



PREDICTING LAUNCH PAD WINDS AT
THE KENNEDY SPACE CENTER WITH
A NEURAL NETWORK MODEL

THESIS

Steven J. Storch, Captain, USAF
AFIT/GM/ENP/99M-11

19990402 015

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

Approved for public release; distribution unlimited.

The views expressed in this thesis are those of the author, and do not reflect the official policy or position of the Department of Defense, or the U.S. Government.

AFIT/GM/ENP/99M-11

PREDICTING LAUNCH PAD WINDS AT THE KENNEDY SPACE CENTER
WITH A NEURAL NETWORK MODEL

THESIS

Presented to the Faculty of the Graduate School of Engineering
of the Air Force Institute of Technology Air University
Air Education and Training Command In Partial Fulfillment of the Requirements for the
Degree of Master of Science in Meteorology

Steven J. Storch, B.S.

Captain, USAF

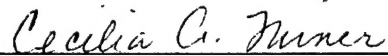
March 1999

Approved for public release; distribution unlimited.

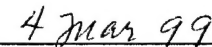
PREDICTING LAUNCH PAD WINDS AT THE KENNEDY SPACE CENTER WITH
A NEURAL NETWORK MODEL

Steven J. Storch, B.S.
Captain, USAF

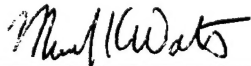
Approved:



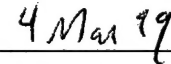
Cecilia A. Miner, Lt Col, USAF
Chairman, Advisory Committee



Date



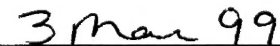
Michael K. Walters, Lt Col, USAF
Member, Advisory Committee



Date



Steven K. Rogers, PhD
Member, Advisory Committee



Date

ACKNOWLEDGEMENTS

I'm indebted to many people who have assisted me in producing this thesis. I thank my advisor, Lt Col Miner, who supported my decision to explore neural network methods. Neural networks were new to both of us, and it took an amount of trust that I would learn enough to complete a useful thesis. Dr. Rogers introduced neural networks and taught me how they work. His encouragement led me to learn neural network programming. Lt Col Walters provided the curiosity and skepticism that made me test and better understand a neural network's solutions. Capt Kelly Greene spent several valuable hours (in the final days of her doctoral dissertation defense preparation) teaching me about recursive neural networks and feature saliency methods, and pointing out invaluable resources. The men and women at the Major Shared Resource Center went out of their way to quickly grant me access to their high performance computers. They were always ready to answer my questions and were conscious of my needs – even during the winter holidays. And thanks to my fellow classmate, Captain Jim Everitt, who traded ideas with me and made the long hours a little more enjoyable.

Steven J. Storch

TABLE OF CONTENTS

	Page
Acknowledgements.....	iii
Table Of Contents	iv
List of Figures.....	vi
List of Tables	viii
Abstract.....	ix
1. Introduction	1
1.1 Overview.....	1
1.2 Background.....	2
1.3 Problem.....	4
1.4 Scope	6
1.5 Approach	6
1.6 Summary of Results.....	6
2. Theoretical Background.....	7
2.1 Chaos Theory.....	7
2.2 Ensemble Forecasting.....	11
2.3 Neural Networks.....	19
3. Methodology.....	27
3.1 Overview.....	27
3.2 Preparing the Observations.....	28
3.3 Developing Predictive Variables	33
3.4 Reducing the Variables.....	33
3.5 Training the Network.....	36

3.6	Generating a Forecast Ensemble.....	37
4.	Results.....	39
4.1	Overview.....	39
4.2	Recursive Neural Network Performance	39
4.3	Ensemble Model Performance	43
5.	Conclusions and Recommendations.....	45
5.1	Conclusions.....	45
5.2	Recommendations.....	46
Appendix A. Original Model Inputs.....		48
Appendix B. MATLAB Code		52
Appendix C. Ensemble Model Fortran Code		53
Bibliography		58
Vita		60

LIST OF FIGURES

Figure	Page
Figure 1 WINDS tower network.....	5
Figure 2 A simple phase space	8
Figure 3 A Lorenz attractor	9
Figure 4 Example of possible phase evolutions	12
Figure 5 Creating a growing mode by adjusting observations	13
Figure 6 Durand-Greville's Synoptic Chart.....	16
Figure 7 Development of Baroclinic and Convective Modes.....	18
Figure 8 Neuron comparison	20
Figure 9 Transfer functions	21
Figure 10 Feed-forward neural network.....	22
Figure 11 Elman recursive neural network.....	23
Figure 12 Error surface with local minimum	24
Figure 13 Example of training to a local minimum.....	25
Figure 14 Stop training method.....	26

Figure 15 Surface buoy observation locations.....	29
Figure 16 Winds tower subset	30
Figure 17 Time-averaged observations.....	32
Figure 18 SSE versus number of variables	34
Figure 19 Basic model structure	37
Figure 20 Ensemble model structure	38
Figure 21 Peak wind performance	40
Figure 22 Model forecast.....	43
Figure 23 Ensemble model divergence.....	44

LIST OF TABLES

Table 1 Surface and Buoy Observation Records.....	29
Table 2 The Predicted Variables	35
Table 3 Ten Additional Predictive Variables	36
Table 4 Mean Absolute Error for a 15-Minute Forecast	41
Table 5 Absolute Error (knots) in Model	42

ABSTRACT

This thesis uses neural networks to forecast winds at the Kennedy Space Center and the Cape Canaveral Air Station launch pads. Variables are developed from WINDS tower observations, surface and buoy observations, and an upper-air sounding. From these variables, a smaller set of predictive inputs is chosen using a signal-to-noise variable screening method. A neural network is then trained to forecast launch pad winds from the inputs. The network forecasts are compared to persistence, and peak wind predictions are found skillful compared to persistence.

An ensemble modeling technique using Toth's and Kalnay's breeding of growing modes method is explored with neural networks. The inputs are perturbed an amount representative of measurement error. Ensemble member forecasts are found to diverge, but the ensemble spread does not often encompass the resulting weather. This is due to a disproportionate amount of error originating from the model compared to error originating from measurements.

1. Introduction

1.1 Overview

This thesis explores the feasibility of numerically modeling winds at the Kennedy Space Center and the Cape Canaveral Air Station (KSC/CCAS) with neural networks. Traditionally, weather phenomena have been numerically modeled with rigorously derived governing equations. However, model shortcomings arise due to an imperfect understanding or representation of those governing equations. Also, typical numerical weather prediction (NWP) models solve millions of variables for many iterations in time and are computationally expensive (Wilks, 1995). Perturbed ensemble forecasts have been generated from the NWP models to approximate model uncertainty, but are also limited by computer speed. Neural networks, however, provide another approach to numerical modeling that is powerful yet computationally efficient. Neural networks don't require an understanding of the governing equations; instead they require past examples of how the weather system behaved. From past examples, neural networks develop approximations of the governing equations with simple parallel linear equations. Their parallelism results in computational speed (Weiss and Kulikowski, 1991). This computational speed may allow one to generate many ensemble solutions from perturbed conditions.

The need to forecast launch pad winds at KSC/CCAS provides a specific problem and impetus to develop a neural network ensemble weather model. Chapter 1 of this thesis discusses the background of the problem and provides a short summary of the problem, scope, approach, and results. Chapter 2 is a discussion of the pertinent theory to this thesis. Chaos theory is discussed first; it is important since the weather is thought to be a chaotic system (Lorenz, 1993).

Chaos theory is the basis of the next section: ensemble forecasting. This section discusses the rationale behind ensemble forecasting and describes a method of generating ensemble member forecasts. Finally, a third section discusses the structure and use of neural networks. Chapter 3 discusses the methodology of how inputs were selected to develop the neural network model and how those inputs were processed. Chapter 4 discusses how well the model performed. Neural network performance and ensemble performance are considered separately. Finally, chapter 5 draws conclusions about the model's performance and makes recommendations for similar modeling attempts.

1.2 Background

Launch Weather Officers (LWOs) at the 45th Weather Squadron (WS) are responsible for providing threshold forecasts that often determine whether or not a space launch proceeds. Each space vehicle, from the Delta II rocket to the Space Shuttle, is designed to tolerate a range of environmental conditions. Likewise, each vehicle requires a unique set of weather thresholds to be met for safe operation. The thresholds include thunderstorm and lightning probability, temperature, and wind speed, to name a few. Launching a space vehicle outside weather thresholds leads to a dangerous situation. A well known example is the space shuttle Challenger, which exploded because its O-rings were not properly designed for cold temperatures (Keel, 1986). The costs of this oversight were enormous in every way. The loss of the Challenger and its seven crew members was a national tragedy, and it moved President Reagan to commission an investigation into its cause. Indeed, it was the greatest setback in NASA's history. The Challenger accident demonstrated the importance of understanding all environmental impacts and launching in a favorable environment. The LWOs of the 45th WS shoulder the responsibility of predicting whether or not environmental conditions will meet the space vehicle's thresholds.

One critical prediction is the launch pad wind forecast. It is important that the winds remain below a threshold level when the shuttle, or a rocket, is standing unsupported on the launch pad. About eight hours before launch during the space shuttle countdown sequence, the structure that supports the shuttle, called the Rotating Support Structure, is rotated away from the shuttle. From that time until launch, the shuttle stands unsupported, but close to the support structure. Winds above threshold, during this critical period, could cause a collision between the shuttle and the support structure. Likewise, all the space vehicles go through a similar sequence of events and have limits to the amount of wind they can safely sustain.

The launch pad wind forecast is one of the important services the LWOs provide and is one of their greatest challenges, especially in winter. Within the critical eight-hour pre-launch window, it is particularly difficult to forecast whether or not the winds will exceed the threshold value, especially when the winds are already close to the threshold value. The causes of strong winds vary by season. During the summer, strong winds are caused by meso-scale events, such as thunderstorms and the sea breeze front. During the winter, strong winds are primarily caused by synoptic-scale events and complicated by meso-scale events and local effects. Changing pressure gradients, fronts, land and ocean temperature contrast, stability, and a complex frictional environment are some of the factors the LWOs must consider. The fact that there are multiple launch pads to consider adds to the complexity of the task.

To assist the LWOs in forecasting the wind, the 45 WS uses a sophisticated weather sensing system to support KSC/CCAS. The Weather Information Network Display System (WINDS) is a collection of 47 meteorological towers placed around KSC/CCAS (Computer Sciences Raytheon, 1998). These towers measure the wind, temperature, moisture and other properties at multiple heights above the surface. The density of the network is about one tower

every 27 km², and is most concentrated around the launch pads. Measurements are recorded at one and five minute intervals. WINDS is one of the most densely instrumented mesonetworks in the United States; it provides the 45 WS with a wealth of current and archived meteorological information.

WINDS often supplies more information than the LWOs are able to absorb and use. In such situations, it's ideal to process the observations into smaller amounts of information the LWOs can readily understand and act upon. Numerical modeling techniques and a body of mathematical theory may be applied to the WINDS observations to do just this. More specifically, neural networks are suited to process complex information into simpler measurements, and chaos theory and ensemble modeling provide information about the likeliness of weather events.

1.3 Problem

The 45 WS has much information about the lower atmospheric winds near KSC/CCAS but lacks a skillful method of applying it towards synoptic wind forecasts. The amount of data is so large, and the complexity of the factors is so great, that the LWOs have not developed successful rules that forecast relatively small changes in the wintertime launch pad winds (Roeder, 1998). Currently, forecasters do not have a consistent method of forecasting changes in the wind on the order of a few knots. This deficiency can become critical when the wind speed is within ten knots of a threshold.

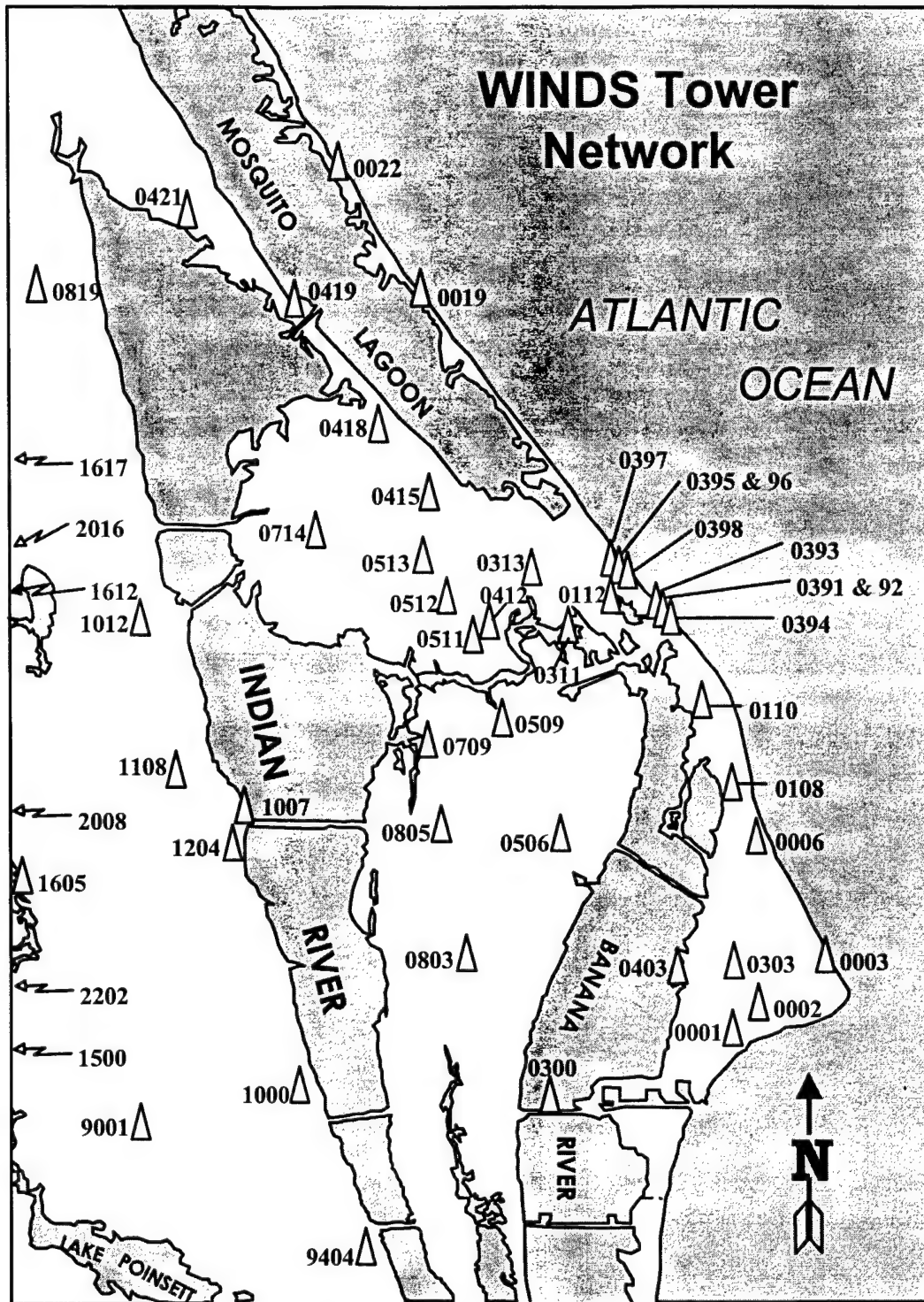


Figure 1 Locations of the 47 WINDS network towers. Most towers record observations at 6 and 54 feet. Towers 0006 and 0110 record at 6, 12, 54, 162, and 204 feet. Tower 0313 records at those levels and 295, 394, and 495 feet; eight levels in all.

Thus, the goal of this thesis is to develop a model for the 45 WS LWOs that improves their ability to correctly forecast launch pad winds. The forecast model must demonstrate predictive power when compared to persistence; that is, the model's forecast must be better than not forecasting at all.

1.4 Scope

This thesis develops a data set containing predictors and outcomes of the launch pad winds and creates a neural network model that forecasts the wind. The model incorporates an ensemble forecasting technique that provides insight into variation caused by measurement errors. The model is validated with previous weather records and compared against persistence to quantify skill.

1.5 Approach

WINDS tower data and other secondary data sources are screened and used to develop the neural network. The network is processed on high performance computers at the Major Shared Resource Center using the MATLAB 5.2 © software package. A Fortran program simulates the network and generates the ensemble forecasts.

1.6 Summary of Results

This thesis develops a neural network model that is skillful against persistence at forecasting the launch pad winds four hours ahead. The model forecast exceeds a persistence forecast for peak wind by as much as 11 percent on average. This thesis also demonstrates an ability to develop perturbed ensemble forecasts, but they do not diverge quickly enough to be representative or useful.

2. Theoretical Background

2.1 Chaos Theory

In 1963, with the release of his paper, "Deterministic Nonperiodic Flow," Edward Lorenz introduced chaos theory. He discussed the implications of observed non-periodicity in natural systems on our ability to predict their future states (Lorenz,1963). Understanding chaos theory is a key element in understanding the weather and the success (or failure) of weather models.

2.1.1 Chaos Discovery

Lorenz described chaos as sensitive dependence on initial conditions. He discovered this dependence while attempting to create a reproducible simulation of the weather with a simple convection model (Lorenz,1993). He created a model with a set of non-linear equations and recorded the output values. Lorenz planned to use the output values to replicate initial conditions and restart his model at any point in time. He thought the model would produce the same outputs as the original model run. During the experiment, Lorenz rounded off the output values to three decimal places, thinking the precision would be sufficient for the model. When he reentered the output as input, he was surprised that the non-linear model produced answers that diverged from the original model run, eventually bearing no resemblance. This divergence resulted from the small perturbation in the initial conditions caused by rounding the input values. Lorenz realized the implications to forecasting the weather. Since there is always some amount of error in our meteorological measurements, even a perfect weather model will eventually diverge from the true atmospheric state. Lorenz stated this in his paper, and launched the field of chaos theory.

2.1.2 Phase Space

Chaos theory is essentially a study of phase space. A phase space is a multidimensional space used to describe the state of a system. Each of the coordinate axes pertains to a variable in the system (Wilks, 1995). A natural system traces a path through phase space (called a phase path, or trajectory) when at least one of its measurements is changing. Figure 2 shows an example of a simple phase space: temperature at some location and time.

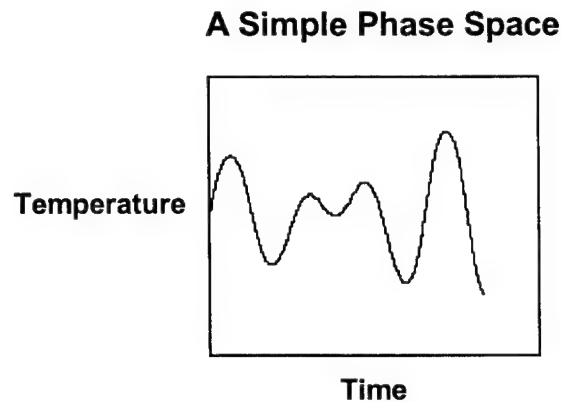


Figure 2 A simple phase space. Temperature and time coordinates define the phase space. The trace of the temperature changing with time is the phase path.

Chaotic systems trace paths through phase space that are bounded, and never cross or repeat themselves (Lorenz, 1963). An example serves well to describe the qualities of chaos. Consider the simultaneous set of equations in the Lorenz convection model:

$$\begin{aligned} dx / dt &= -8/3 x + yz \\ dy / dt &= -10 y + 10 z \\ dz / dt &= -xy + 28 y - z . \end{aligned}$$

These equations trace out the phase path in Figure 3 when started from the initial condition $x = 35$, $y = -10$, and $z = -7$.

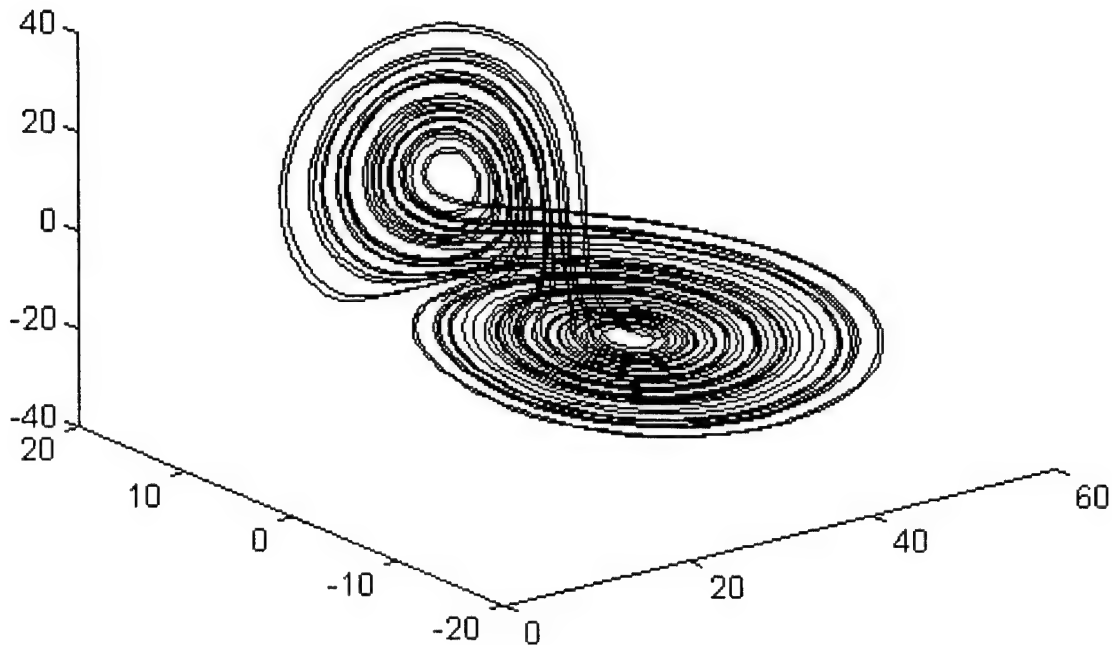


Figure 3 A Lorenz attractor. This figure shows the phase path of the simple convection model developed by Lorenz (1963).

The axes represent the magnitude of the component in the x , y , and z directions. The smooth line connecting continuous points in phase space implies time. The function is bounded, because it repeatedly passes arbitrarily close to at least one point (Lorenz, 1963). In this case, it orbits arbitrarily close to two points. The function does not intersect itself; at no time does it take on an identical state as another time. Another property of this function is it contains a closed set of points. Since the function is bounded, all its solutions exist in a finite area. Within that area, there is a smaller set of points that are solutions to the function. The limit set of points in this, or any other chaotic system, is called an attractor.

2.1.3 *Chaotic Weather*

Our earth's weather meets the requirements to be called a chaotic system. Should one build a phase space with temperature, humidity, and wind measurements from locations around the world, one would most certainly find that it's been a bounded system. In fact, this phase space has been developed in the field of climatology. A large pursuit of climatologists is to describe the bounds of the states the atmosphere holds. The requirement that the earth's weather should never repeat itself to be considered chaotic also holds true. If measured with enough phase dimensions, the earth's weather is found to have never held two identical states.

The so-called dishpan experiments suggest the primary forcings that govern the earth's chaotic weather. The dynamicist Dave Fultz built a model of the atmosphere in the early 1950s using a pan of water on a turntable (Hess, 1959). The pan was heated at the edges to simulate the tropical heating and cooled in the center to simulate polar cooling. The experiments produced jet streams and temperature patterns that look remarkably like the earth's weather. Some of the larger models even produced frontal boundaries (Holton, 1992). The experiments indicated differential heating, gravity, and the earth's rotation are the primary forcing mechanisms that control our weather (Lorenz, 1993).

Differential heating, gravity, and the earth's rotation dictate certain states the atmosphere can hold. Heating air in one location more than another creates a difference in air density and causes circulations due to gravity. The earth's rotation modifies the circulations through the Coriolis force. Much of the atmosphere's motions can be explained through these three forcings, and they often work together as a stable system; that is, they don't enter positive feedback states where disturbances amplify. Fair-weather cumulus development is an example of a common atmospheric process that damps out. Differential heating at the surface often results in cumulus

clouds, but the increased cloud cover often acts to reduce differential heating. Gravity waves are another example. When a mountain disturbs airflow, the waves in the disturbed flow are initially strong, but generally damp relatively quickly as they move away from the mountain. The point is, for many weather processes, differential heating, gravity, and the earth's rotation work together in such a way that tends to return the atmosphere to a stable condition.

There are unstable conditions of the atmosphere that do develop, however. Otherwise the weather could be predicted much more accurately. When baroclinic systems are numerically modeled, small errors amplify quickly and cause significant errors over a period of days.

With this knowledge, one can address the problem at the Kennedy Space Center. During summer, weather disturbances tend to be gravitational, or due to convection. The weather phenomena that occur are primarily due to the differential heating of the land and ocean. This causes a dense air over buoyant air situation, and the gravitational result is thunderstorms. Such thunderstorms and other gravitationally induced disturbances are stable modes of the atmosphere, and relatively easy to forecast. In fact, near the Kennedy Space Center, thunderstorms occur about two out of three days in the early afternoon every summer. In the winter, however, unstable modes appear. The baroclinic instabilities are enhanced by the strong contrast between the land and sea temperatures. The baroclinic modes present challenging forecasts.

2.2 Ensemble Forecasting

One of the more intriguing concepts to occur in dynamic meteorology is ensemble modeling. An ensemble model, in the meteorological sense, is a collection of model runs for the same time period. Usually the ensemble members are derived from slightly perturbed initial model conditions. The premise is that due to measurement error, one can't know the initial

conditions exactly. No matter how small the measurement error is, it will eventually lead to large errors. This is in part because some of the error is fast growing error that contributes to the atmosphere's baroclinicity. An ensemble forecast seeks to simulate the fast growing errors that could be part of the measurement error. A good ensemble of forecasts relies on two things: 1) The model must be a good model of the attractor, and 2) The perturbations must be close approximations of the maximum growing error (or growing mode), both in size, and in direction in phase space (Toth and Kalnay, 1993).

The reason the error must be similar in size and direction can best be seen in a simple phase space. Consider a one-dimensional phase space of temperature at some location as it evolves over time, as depicted schematically in Figure 4. The dark line represents the evolution of the measured temperature in phase space. The other lines represent nearby trajectories of the

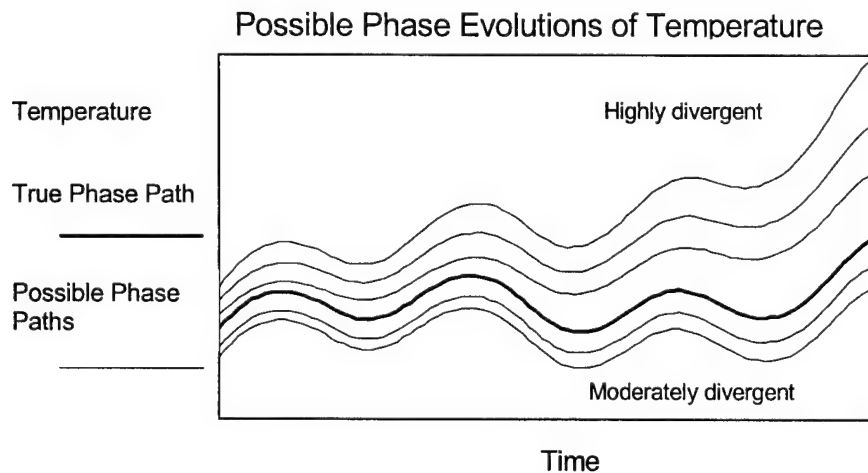


Figure 4 This schematic depicts the temperature changing with time. The fine lines are nearby trajectories in phase space. In this situation, the phase paths diverge more quickly for higher initial values of temperature.

atmospheric attractor; they are possible states the atmosphere may hold. In this case, perturbing the temperature by warming it slightly causes the solution to diverge faster than by cooling it the same amount. In a multidimensional phase space, the direction of perturbation for each dimension (each scalar measurement) is chosen so the errors grow the fastest.

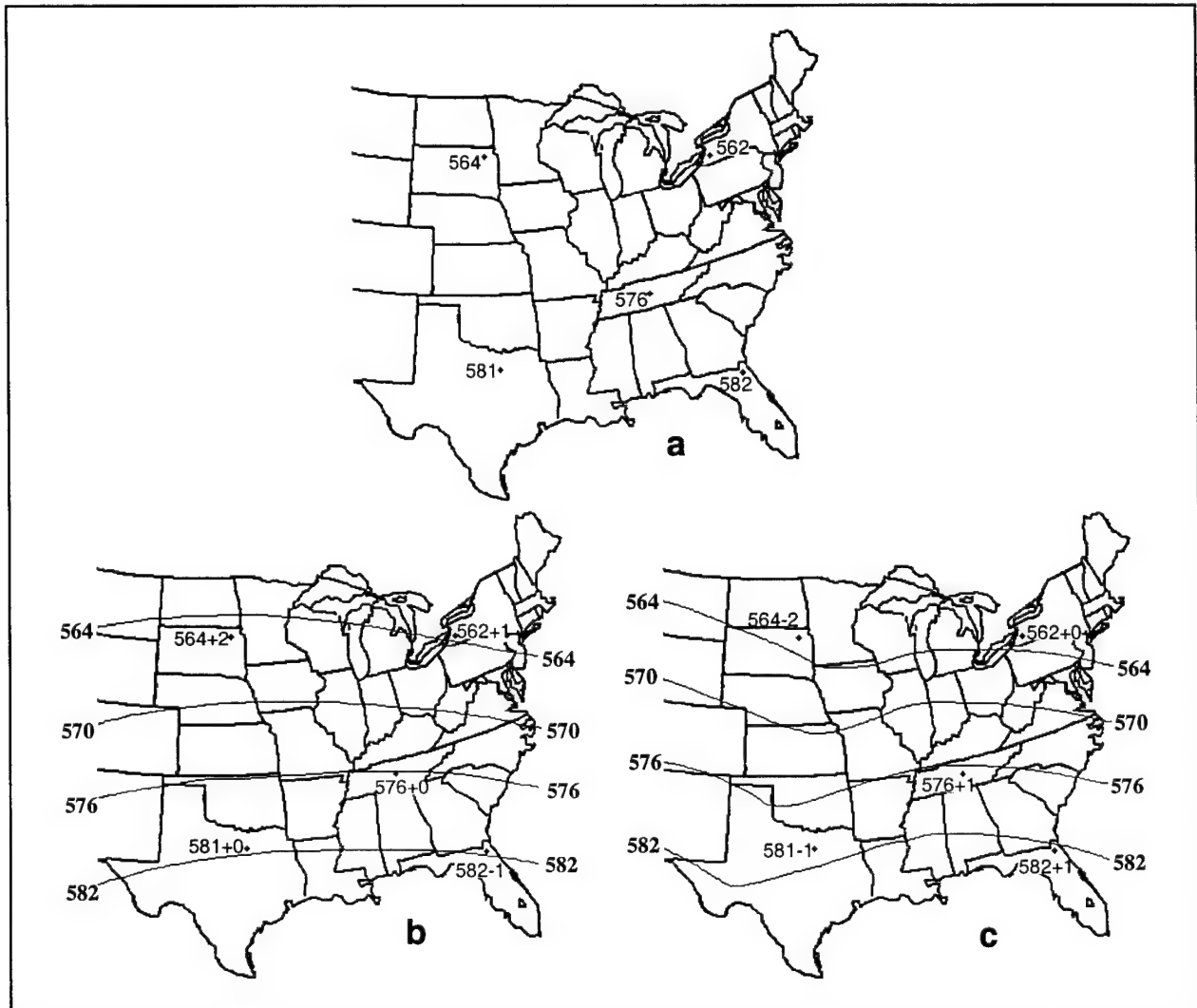


Figure 5 This set of figures demonstrates how a baroclinic growing mode of the atmosphere can be interpreted from measurement error. Five 500 mb height observations are given in chart a, each with up to 20 meters error. The error could be adjusted into the benign pattern in chart b, or into the baroclinic pattern in chart c. Both are plausible interpretations, given the limits of the data.

To understand how errors can be organized into a growing mode of the atmosphere, consider that synoptic forecasters have been doing just this for over one hundred years. They do it by using their imagination, weighting certain stations more strongly than others, and fitting a pattern to the measurements. When a synoptic forecaster first looks at plotted surface chart, it has measurement errors in it. The forecaster often adjusts the data to create the weather pattern believed to exist. Figure 5 is a demonstration. Suppose there are only five upper-air observations over the Central and Eastern United States at 500 mb (Figure 5a). The margin of error for each measurement is considerable, say ± 20 meters. By adjusting the observations within the margin of measurement error, the height field can be analyzed in a relatively non-developmental pattern, as in Figure 5b. Or, by adjusting the error in the data another way, it can be analyzed as a weather pattern that is more likely to develop, as in Figure 5c. There are many ways to adjust the error in the data that produce physically allowable, and thus possible, states of the atmosphere.

One way to explore and use the measurement error in the atmospheric analysis is through phase space. Each measurement for each location is assigned a phase dimension. In the example above, the observations of 500 mb heights produce the five-dimensional phase space. At a moment in time, it can be represented by the five-dimensional vector **A**, where each measurement holds a position in the vector:

$$\mathbf{A} = 5640 \quad 5620 \quad 5760 \quad 5810 \quad 5820$$

In this case, the South Dakota observation holds the first dimension, the New York observation holds the second dimension, and so forth. The measurements contain an amount of error represented by the vector **E**. In this case, the error is ± 20 meters of the measured value. Then the bounds of **E** are

$$\mathbf{E}_{(bounds)} = \pm 20 \quad \pm 20 \quad \pm 20 \quad \pm 20 \quad \pm 20.$$

The true value of the error is, of course, unknown, so it can be used to adjust the analysis **A** as is beneficial to describing the true state of the atmosphere. In Figure 5b, the error corresponding to each dimension was assumed to be

$$\mathbf{E} = 20 \quad 10 \quad 0 \quad 0 \quad -10,$$

while in Figure 5c, different error was assumed:

$$\mathbf{E} = -20 \quad 0 \quad 10 \quad -10 \quad 10.$$

Both assumptions are allowable because they fall within the bounds of measurement error. The difference is, if the errors in the first case are true, a rather benign weather regime exists, and the weather will change little. But if the errors in the second case are true, a baroclinic wave is generating, and the weather will change dramatically. The second case represents a maximum growing mode of the atmosphere. It's a way the measurement errors can be organized to cause the maximum amount of development, or change from the current state. From a forecaster's stand-point, it's good to know the maximum amount of development one could expect.

The work of past synoptic forecasters is impressive and demonstrates the ability of the human mind to organize measurement errors into useful patterns. In 1892, Durand-Greville drew a beautifully organized synoptic surface chart depicting squall line passage over Europe, 27 August 1890 (Figure 6; Fujita, 1986). It was no small effort to gather data from across Europe in the 1890's, and Durand-Greville must have compensated for many errors resulting from imprecise measurements, varying times of measurement, and lack of any measurement at all. Yet his chart looks like one a good forecaster might draw today. He did it by adjusting the

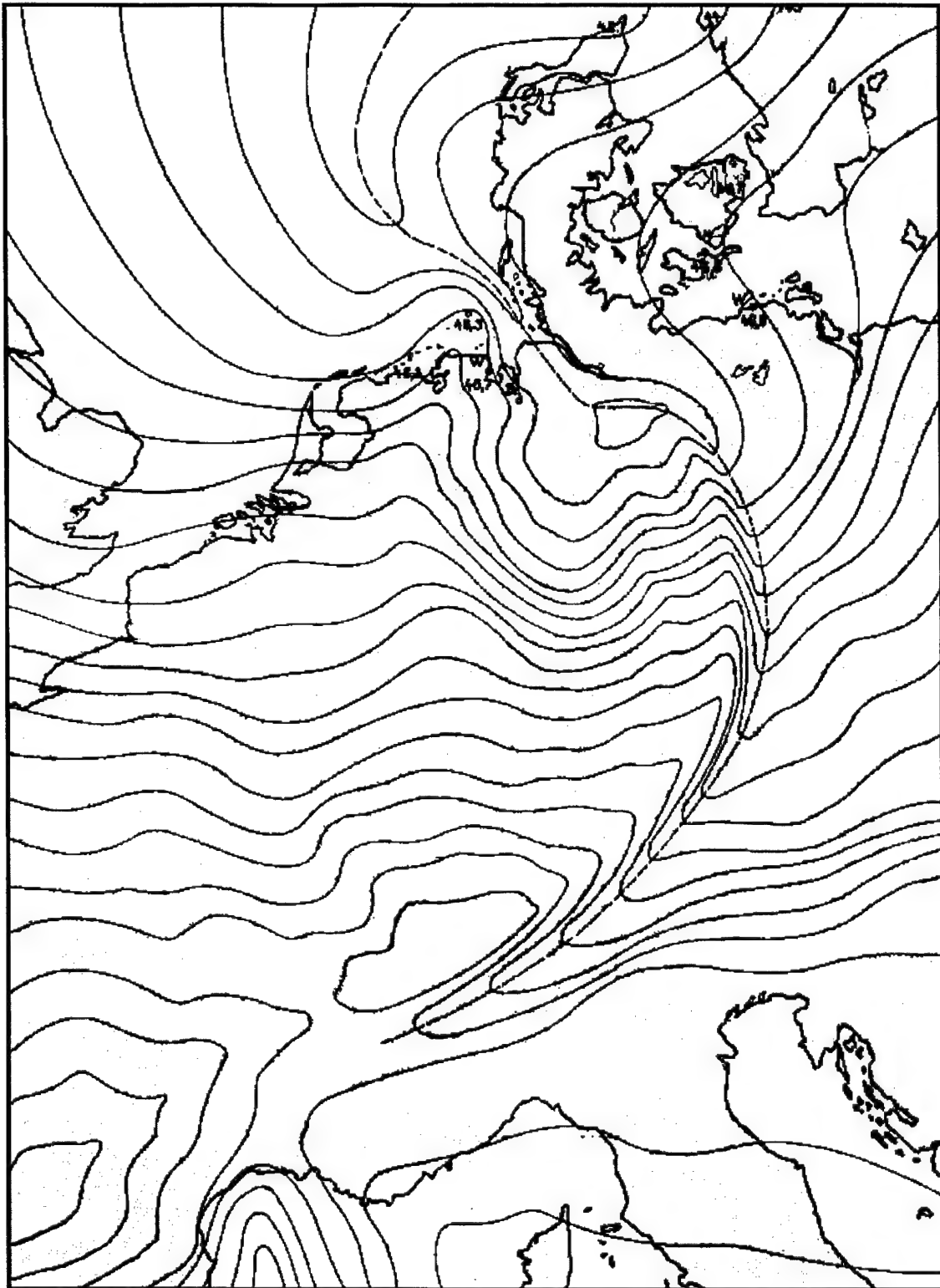


Figure 6 Recreation of Durand-Greville's synoptic chart, completed in 1892. The chart shows the pressure field of a squall line across Europe. Each lines represents an isobar of 1 mm of mercury. Durand-Greville's chart demonstrates that forecasters organize missing data and measurement uncertainties into reasonable weather patterns that develop and cause weather. Adapted from Fujita,1986.

measurements within what he thought were the margin of error.

Forecasters can adjust the measurement errors in an analysis to produce realistic atmospheric patterns, but can a computer do the same? Can a computer mimic this forecasting art? The surprising answer is yes, and it is done in ensemble forecasting.

In the early 1990s, Toth and Kalnay developed a simple and accurate method of simulating growing modes. They called it “breeding of growing modes,” or BGM (Toth and Kalnay, 1993). To use this method, one first estimates the magnitude of the average error vector in phase space. This effectively estimates the error bounds for each measurement in the analysis. Next, the initial conditions of a model run are perturbed with a random error vector. The magnitude of the error vector is set equal to that of the average error vector. At first, the perturbed model usually converges towards the control forecast. This is because most of the error contributes in ways that are physically improbable. The model quickly regains balance by developing gravity waves and through other processes (as would the true atmosphere). But some of the initial random error contributes in physically meaningful ways that grow and develop. These growing errors tend to be errors that contribute to the baroclinicity of the atmosphere. The baroclinic part of the error catches up to and soon exceeds the gravitational (or convective) part of the error, as depicted in Figure 7. After sufficient time, the baroclinic errors dominate the error vector, and the solution diverges from the control forecast. At the end of the forecast period, the fast-growing error vector is scaled back to the size of the average error vector. The scaled-back fast-growing error vector is then used to perturb the next set of initial conditions for the control forecast. This cycle repeats, and soon the fast-growing error vector is a good approximation of a maximum growing mode of the atmosphere. The growing mode can be recycled indefinitely.

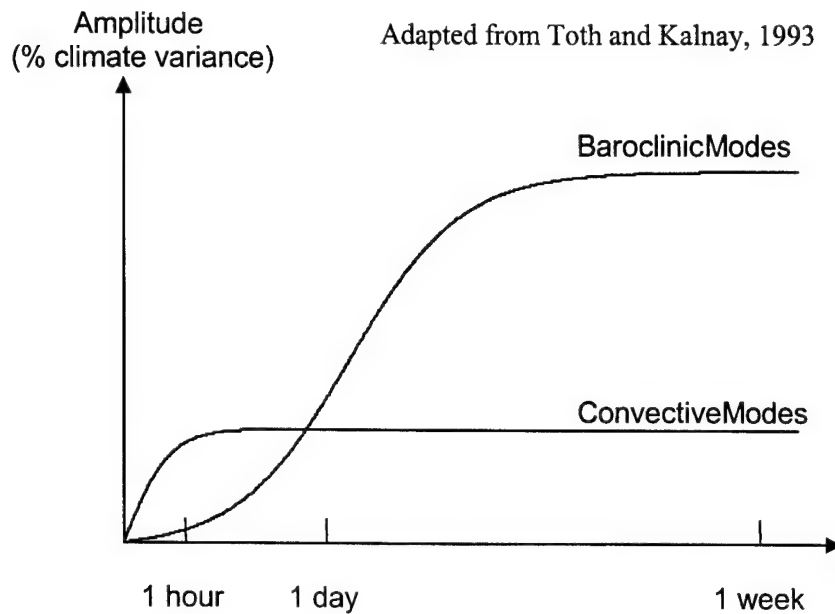


Figure 7 A schematic of the development of baroclinic and convective modes of the atmosphere. Convective instability causes more variance in the atmosphere on a time scale of a few hours. For longer time scales, baroclinic instabilities account for more of the variance.

Multiple growing modes can be generated from other random perturbations. Each growing mode is slightly different, just as synoptic forecasters' interpretations and drawings of a weather chart differ in opinion to opinion. The different growing modes develop into an ensemble of model forecasts. The forecasts can be averaged, and the ensemble average usually predicts the weather better than any individual model forecast (Wilks, 1995).

So far, developing ensemble models with growing modes has been somewhat of a computationally expensive process. A weather model must recalculate all its solutions for each ensemble member. Due to the size of most weather models, ensemble modeling has been limited to centers with large computers. However, there are simpler methods to model the weather that may allow others to explore the power of ensemble modeling. Neural networks provide one of those methods.

2.3 Neural Networks

Neural networks have long been used for time-series prediction of chaotic systems and systems with unknown governing equations. In 1964, Hu used Widrow's adaptive linear network to forecast the weather (Gershenfeld and Weigend, 1994). In the 1992 *Santa Fe Time Series Prediction and Analysis Competition*, Wan placed first using a neural network to predict the Lorenz equations over a relatively short-term of 100 time steps.

However, neural networks have been slow to gain popularity in meteorology due to the relative shortness of many data records and the large spacial fields that must be dealt with (Hsieh and Tang, 1998). Low spacial or temporal resolution may cause nonlinear instabilities in a neural network due to overfitting. Hsieh and Tang state that these obstacles may be overcome by using nonconvergent training methods and prefiltering the data, as will be explained.

Neural networks began as an attempt to build numerical models that work in a similar manner as the brain (Gershenfeld, 1999). Figure 8 compares a biological neuron to an artificial neuron. In the brain, signals are received through synapses located on dendrites (Haykin, 1994). The dendrites transmit the signals to the neuron center (the soma) by means of electrical potential. The neuron sums potential received from all dendrites, and if the potential is large enough, it activates and transmits an electric impulse across the axon. This electrical impulse is the signal to other neurons. Similarly, an artificial neuron, called a node, collects signals modified by a network of weights. The signals are summed in the node, and processed through a function to "activate" the sum into a new signal. The signal is then transmitted to other nodes.

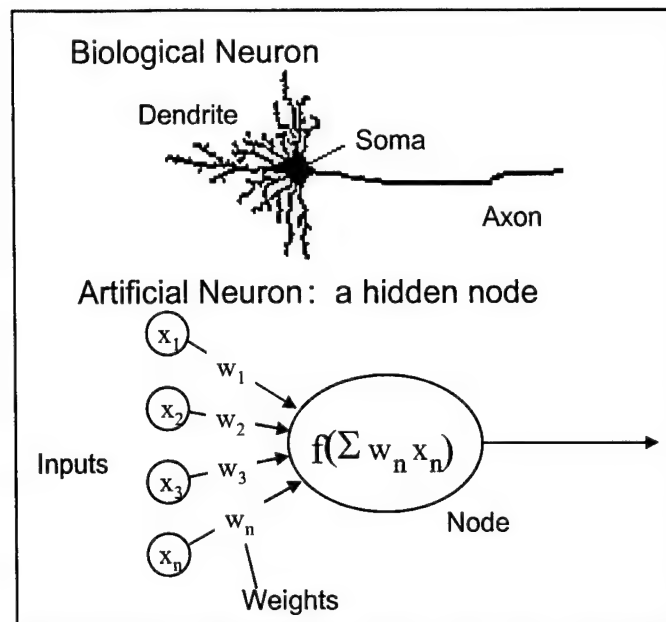


Figure 8 An artificial neuron is designed after its biological counterpart. Where a biological neuron receives information through dendrites, an artificial neuron receives information through a set of weights. Each activates when the combined signals exceed a threshold.

Neural networks process signals in parallel through layers of interconnected nodes. Each of the j nodes, y_j , is a function of the summation of each of the i input terms multiplied by a weight, and a bias. This relationship is expressed as follows:

$$y_j = f\left(\sum_i w_{ij}x_i + b_j\right)$$

where x_i is the input term, w_{ij} is the weight of x_i to node y_j , and b_j is the bias to node y_j . The function f (called a transfer function) is chosen for the characteristics of its solutions. Figure 9 displays three prominent transfer functions. The hyperbolic tangent is a useful transfer function. It's sometimes called a squashing function for its ability to "squash" an input between -1 and 1 . These positive and negative signals provide a kind of "on/off" information similar to the way the brain sends electrical impulses through its neurons. Another common transfer function is the logistic function. It is similar to the hyperbolic tangent, but bounds its solution between zero and one. The linear function allows an unbounded solution.

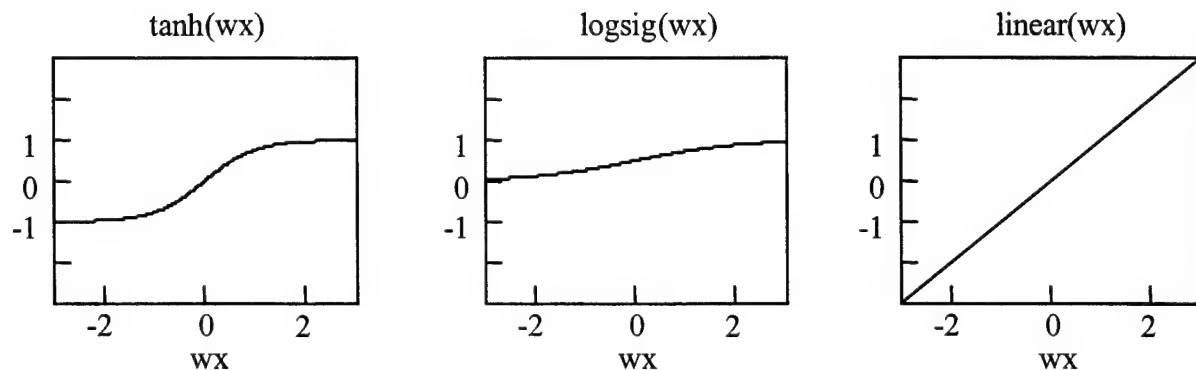


Figure 9 Three common transfer functions. The hyperbolic tangent gives values near 1 when it is active ($y = \sum wx + b$ is positive) and values near -1 when it's not active. The logistic sigmoid is similar, but gives values near 0 when it's inactive. The linear function doesn't restrict the size of the solution.

Neural networks are perhaps best described with images. Figure 10 is a diagram of a two-layer feed-forward neural network. The input layer is denoted by x variables, the middle layer by y variables, and the output layer by z variables. Inputs are multiplied by a weight, w_{ij} , represented by the arrows. The bias represents a constant unit signal multiplied by a weight. Each weighted input and the bias are summed in the $n1$ layer. Since the values in this layer are transitory and generally unimportant except to arrive at the output, it is called a hidden layer (likewise, its nodes are called hidden nodes). Each $n1$ is processed through an activation function to become the hidden node value. The cycle repeats with the hidden node layer (the y variables) providing inputs to the output layer (the z variables).

Neural networks can be very powerful. By using a linear equation for the transfer function to the outputs, the outputs can take on any value. This is because the linear function allows unbounded solutions. When the hyperbolic tangent or logistic transfer functions are used in the hidden layers, the network is able to interpret non-linear signals. These two abilities give neural networks their strength of generalization. A two layer feed-forward neural network with enough hidden nodes, a hyperbolic tangent activation function, and a linear final transfer function can be trained to approximate any continuous function arbitrarily well (Demuth and Beale, 1998). That is, with enough data and a sufficient number of nodes, a network can

simulate any function as well as desired. This ability to generalize is a neural network's capital strength.

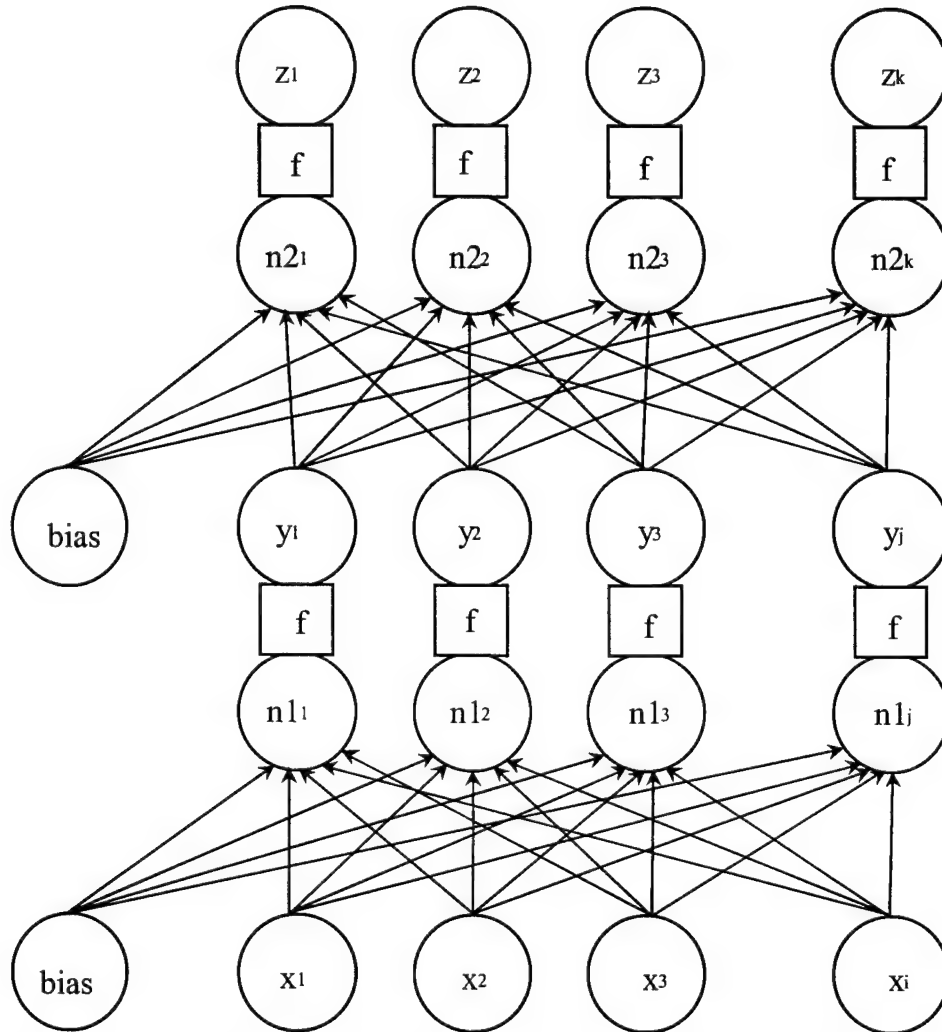


Figure 10 A representation of a neural network. Inputs are denoted by x values, hidden nodes by y values, and outputs by z values. The inputs and bias are weighted and summed (represented by arrows leading to the $n1$ units). These sums are modified by a transfer function, f , and the results are hidden node values, y . These hidden node values are processed likewise to obtain the outputs, z .

Another type of network, recursive neural networks, often can outperform feed-forward networks by simulating memory. For this property they were selected for use in this thesis. These networks consider not only present inputs, but past inputs as well. An example is the Elman recursive neural network. It simulates memory by feeding its previous hidden nodes to its

present hidden nodes as inputs (Figure 11). The recurrent nodes allow the Elman to both detect and replicate time-varying patterns (Demuth and Beale, 1998). The Elman's memory may last four to six time steps before the past signals are lost in the data's noise (Greene, 1998). Another type of network, the adaptive time delay neural network, adapts time delays to different variables to gain the best predictive signal (Gainey, 1993).

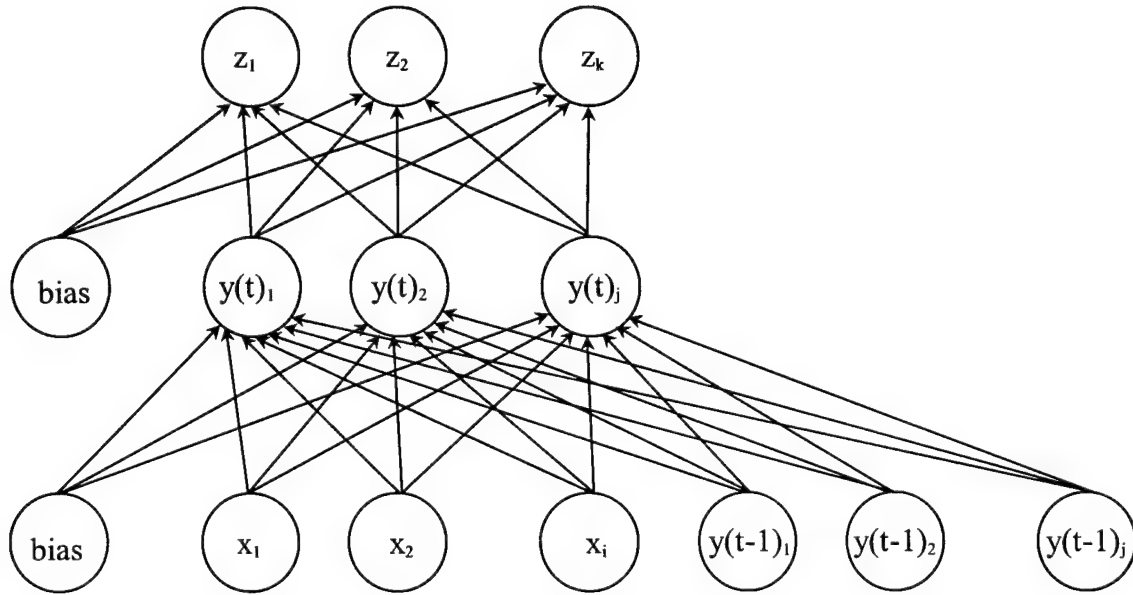


Figure 11 The Elman recursive neural network simulates memory by using hidden node values from the previous time step as inputs.

Neural networks learn to approximate a function by reducing the error between computed outputs (z_k) and the given values of the outputs (t_k). Normally, the network does this by reducing the sum of squared error (SSE) over n exemplar cases:

$$SSE = \frac{1}{2} \left(\sum_n \sum_k (z_{nk} - t_{nk})^2 \right).$$

One method to reduce the sum of squared error is through gradient descent (Gershenfeld, 1999).

The gradient descent method determines the slope of the error field (Δw_{ij}) and corrects the

weights by differentiation:

$$\Delta \mathbf{w}_{ij} = -\eta \frac{\delta(SSE)}{\delta \mathbf{w}_{ij}}$$

where η controls the size of the correction increment (the learning rate). Typically, the network considers all the training cases (called a training epoch), and then corrects the set of weights by $\Delta \mathbf{w}_{ij}$ to decrease the sum of squared error. As long as the transfer functions are continuously differentiable, the error gradient slope can be determined. Generally, the network will search for the point in the error hypersurface that is closest to zero.

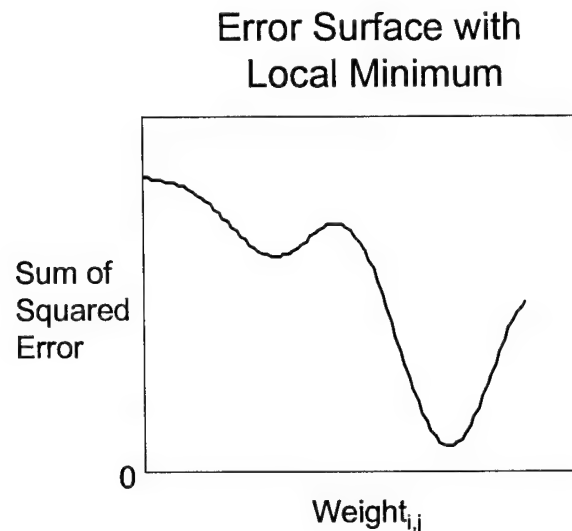


Figure 12 A local minimum in the error surface may prevent a neural network from finding the global minimum.

One of the challenges of designing a network is to overcome local minima in the error hypersurface. As the network descends the error gradient, it sometimes becomes trapped in a local minimum, such as in Figure 12. These minima seem to occur when the network approximates a function that is similar to the actual forcing function, but altered by some constant, or set of constants. Figure 13 shows outputs of two neural networks, both trained on a

set of points generated from the Lorenz equations presented earlier. The first network was approaching the global minimum of the data set; it looks similar to the actual attractor (depicted in Figure 3). The second trained to a local minimum; it appears quite warped.

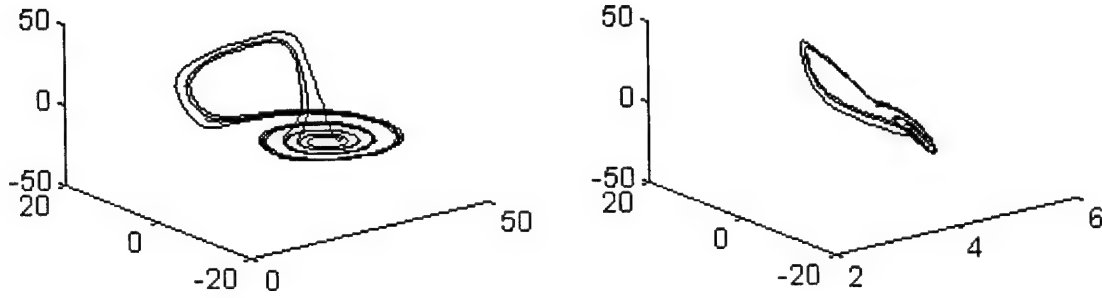


Figure 13 The two figures are simulations from separate feed-forward neural networks trained with the same data from the Lorenz equations. The simulation on the left is from a network that trained near the global minimum. The simulation on the right trained to a local minimum.

Convergence to local minima can be often be cured by adding a momentum term to the learning rate. One does this by considering the past slope of the error surface ($\Delta \mathbf{w}_{ij}(t-1)$) in the slope of the present error surface ($\Delta \mathbf{w}_{ij}(t)$):

$$\Delta \mathbf{w}_{ij}(t) = -\eta \frac{\partial (SSE)}{\partial \mathbf{w}_{ij}(t)} + m(\Delta \mathbf{w}_{ij}(t-1))$$

where m is the momentum constant. With enough momentum, the network will roll out of the local minima. With too much momentum, it will become unstable and unable to settle in the global minimum. Typical values of momentum are chosen near 0.9 (Weiss and Kulikowski, 1991), but the best value is dependent upon the data and found by trial-and-error.

Noisy data presents another challenge that must be overcome. Nearly all data has measurement error because measuring devices are designed to operate within an error tolerance. Measurement errors teach the network a false impression of the actual forcing function. If a network is allowed, it will train on a data set until it learns the false signal in the noise. At this

point the network has overfit the data and it begins to lose its ability to generalize the forcing function.

One solution to overfitting is the stop-training method. Stop-training is a nonconvergent method that works by comparing two sets of data that adhere to the same forcing function. First, a validation set of data is removed from the training set. The network trains on the training set, and tests its solution on the validation set. As long as the SSE for the validation set decreases, the network continues training (Figure 14). When additional training causes the validation set's SSE to increase, the network stops training and returns to the optimum weights and biases.

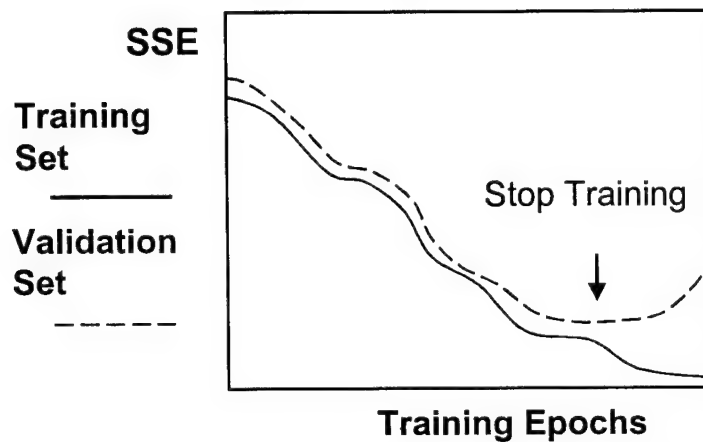


Figure 14 The stop-training method prevents overfitting by halting training when decreasing the sum of squared error in the training set increases the sum of squared error in the validation set.

Data preparation techniques such as normalization and prefiltering also improve convergence and a neural network's ability to simulate a function. Normalizing data within the bounds of -1 or 0 to 1 often improves training performance (Weiss and Kulikowski, 1991). Prefiltering methods such as principle component analysis or canonical correlation analysis may be used to reduce the number of input variables (Hsieh and Tang, 1998). The purpose of prefiltering data is to remove variables that obscure the predictive signal in the data set.

Greene (1998) developed a signal-to-noise variable reduction technique as a prefiltering method. To use this technique, one generates a string of random numbers and includes it in the training set of data. This random variable represents pure noise. A network is trained on the inputs, and the sums of the squared weights for each variable are compared to those of the random variable. The variable with the lowest signal-to-noise ratio is removed from the training set, and the network is trained again. This process is continued until all variables have been removed. Upon completion, the set of variables that performed best (lowest SSE) is chosen as the most predictive.

3. Methodology

3.1 Overview

In this study of the launch pad winds there were five steps taken to develop the neural network model. First, the weather observations were selected and prepared as continuous scalar weather variables. Second, performing vector subtractions developed variables representing a measurement change over some distance (“deltas”). Third, the most predictive variables were chosen using the signal-to-noise variable screening method. Fourth, an Elman recursive neural network was trained on the data. Finally, an ensemble of forecasts was developed using a Fortran program to simulate the neural network.

3.2 Preparing the Observations

A successful neural-network model requires good data. The network can be trained only to the quality of its data. Data preparation is the most important and most time-consuming step

of the process. It includes choosing the data, accounting for missing values, and fitting the data to one time scale.

3.2.1 *Choosing the Weather Observations*

The first requirement to build this neural network model is to choose relevant data. For best results, one must choose high-quality, *predictive* weather variables (input) that explain much of the variation in the *predicted* wind variables (output). Data was collected from scales larger than the WINDS network to provide the neural network with information about the large-scale environment. The collection of weather observations fit into three spacial and temporal scales. The WINDS towers were spaced on the order of a mile and recorded observations each five minutes. Surface and buoy observations were spaced on the order of 100 miles and recorded observations each hour. The Tampa sounding was the only upper air observation and was recorded each twelve hours.

In general, observations were selected for the quality of their data and location of their observation site. Observation data sets with a high percentage of missing values were discarded, and sites were selected to present a balanced representation of the region.

The surface and buoy observations were screened first for large blocks of missing data. Thirteen surface stations from the Florida Peninsula, three Coastal-Marine Automated Network (C-MAN) stations, and two buoys in the nearby Atlantic waters were considered. Off-hour reports were stripped from the observation sets. A total of five surface stations, a C-MAN station, and a buoy were then selected for their completeness of hourly data records (Table 1) and location (Figure 15).

Table 1 Surface and Buoy Observation Records

Station	Type	% Observations Available
Daytona	Surface Report	95.1
Jacksonville	Surface Report	96.6
Melbourne	Surface Report	93.6
Tamp	Surface Report	96.5
W. Palmbeach	Surface Report	98.1
41009	Buoy Station	96.3
SPGF1	C-MAN Station	98.8

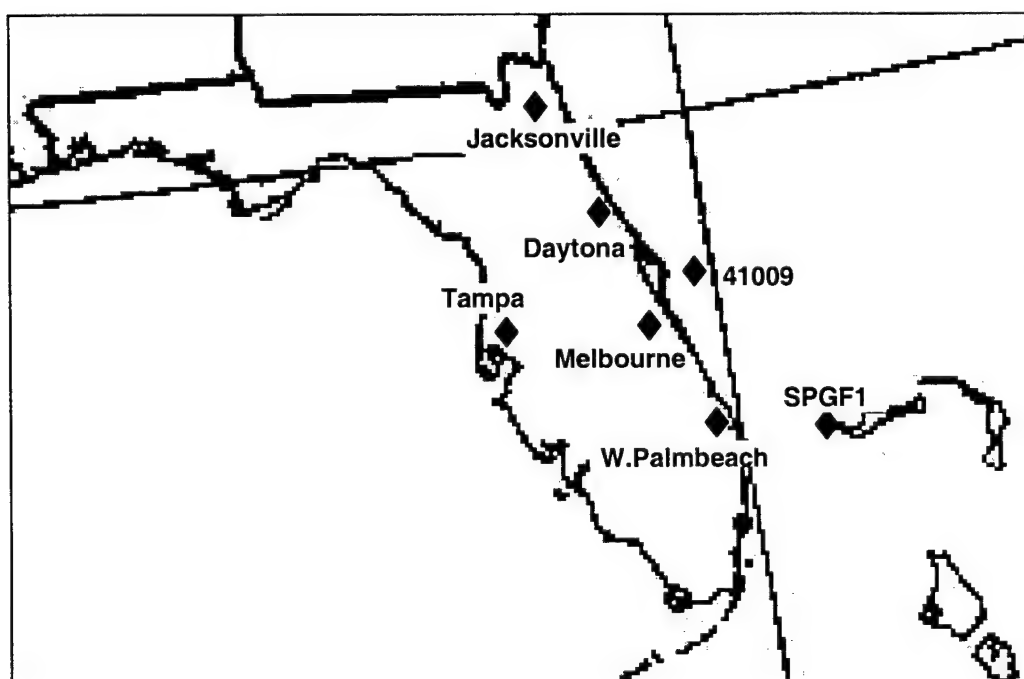


Figure 15 Surface and buoy observation locations. The upper-air observations from Tampa were also used.

The WINDS tower data set was also screened in a similar method to obtain the more manageable subset depicted in Figure 16. All the towers at launch pads were initially chosen. In the case that a launch pad had two towers, the tower with westerly sensors was kept (presuming

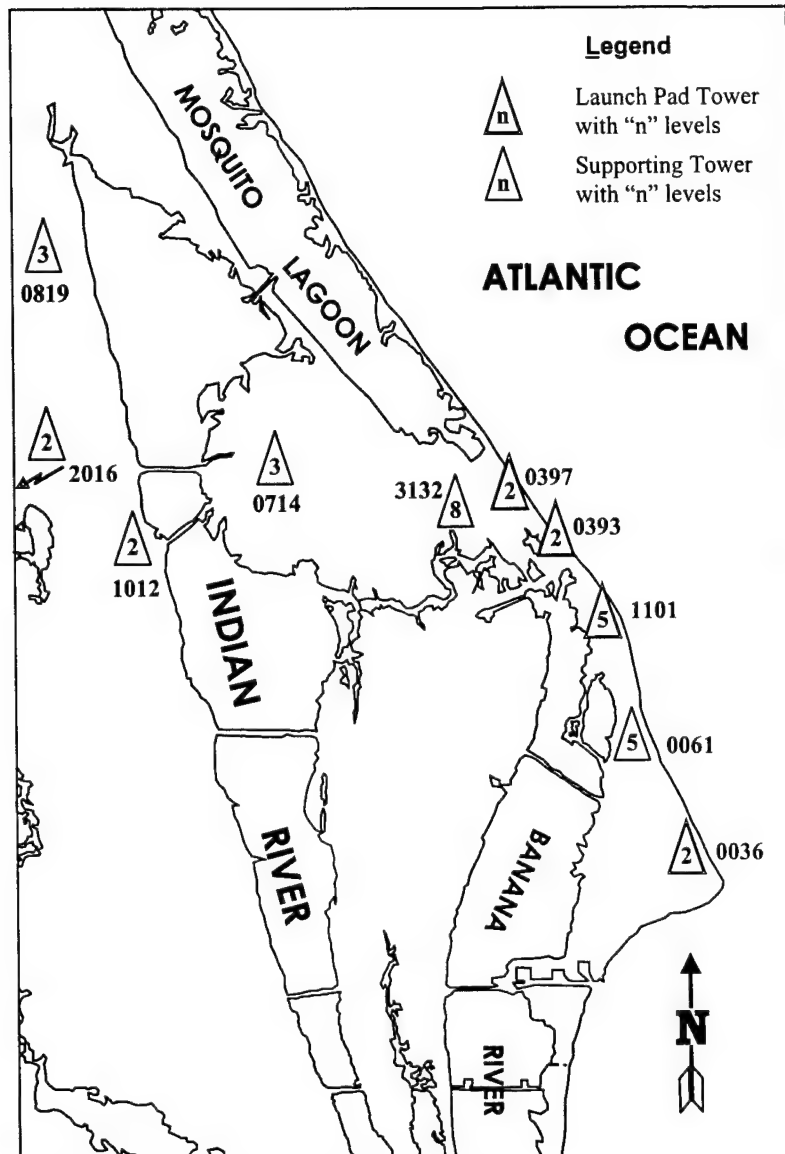


Figure 16 Ten towers used to develop weather variables predictive of the launch pad winds.

most baroclinic systems would cause winds with a westerly component). The launch pad tower 0002 was removed from the set because of an incomplete record due to late construction of the tower and other extended outages as well. The next type of tower selected included towers with three or more sensor levels. These towers tended to be in the vicinity of the launch pads, had high maintenance priority, and relatively complete data records. Several two-level towers on the

far perimeters of the WINDS tower network were chosen for their upstream location to the generally southeasterly travelling baroclinic systems. Completeness of data record was considered by examining the time series of observations on a MathCad© graph for obvious discontinuities.

3.2.2 *Replacing Missing Values*

Replacing missing values in the data set is a problem often encountered while preparing a neural network, as it was in this thesis. Tsoukalas and Uhrig (1997) cite the common sense (and technically correct) thing to do is replace missing values with the best estimate of what they would have been. A number of sophisticated methods of estimating missing values exist (Bishop, 1995), but a linear function fit over the period of missing data was chosen to make a simple estimate of their values. This type of estimation was used on data set E (astrophysical data from a variable star) in the Sante Fe time series prediction contest (Gershenfeld and Weigend, 1994), and its advantage is simplicity and relative accuracy for short time periods.

3.2.3 *Fitting Observations to One Time Scale*

Since the observations were taken at different time intervals, all observations were fit to one time scale. To train the network on realistic values, observations were arranged in time as they normally become available to forecasters. Upper-air observations were assumed to become available 90 minutes after the observations time. Surface observations were assumed to become available on the hour. WINDS tower observations were assumed immediately available.

To eliminate some of the high frequency waves that may be considered noise to the WINDS network, a simple one-dimensional three-point smoother was applied to the observations (Haltiner and Williams, 1980). The smoothing function, f_j , is described as:

$$\overline{f_j} = (1 - S)f_j + \frac{S}{2}(f_{j+1} + f_{j-1})$$

where $S = 2/3$. The smoother dampened waves with frequencies greater than ten minutes. Since information within the ten-minute frequency band was eliminated, the data set was down-sampled by choosing each third observation. This effectively averaged each group of three consecutive observations. Figure 17 is an example of a WINDS tower data segment before and after smoothing and demonstrate some of the strengths and weaknesses of this technique. Though the time-averaged signal is smoother, it understates the actual peak wind by two knots.

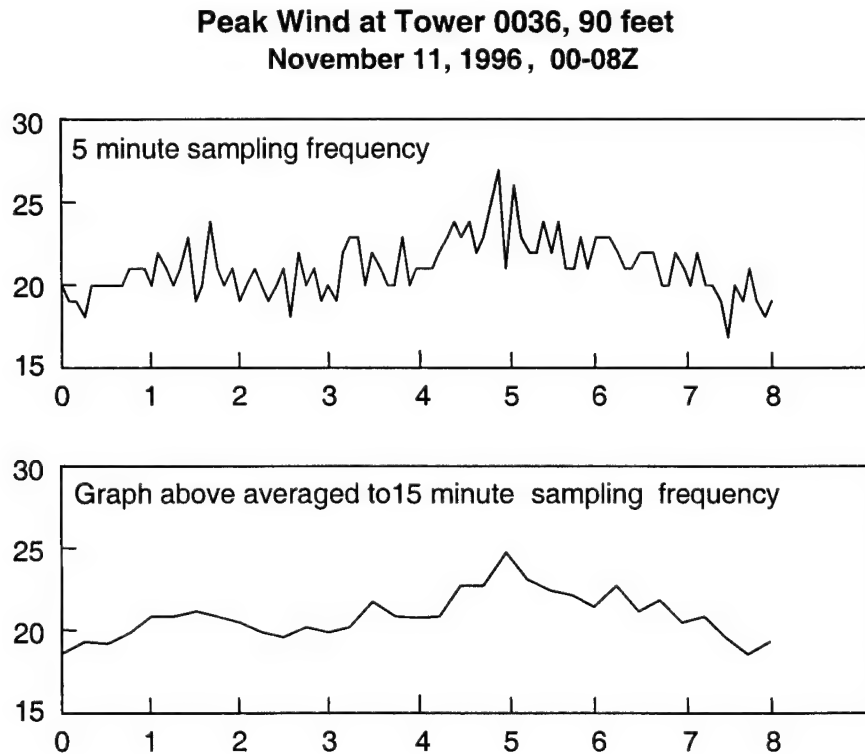


Figure 17 The WINDS tower observations were smoothed by averaging three consecutive points.

3.3 *Developing Predictive Variables*

Wind measurements dominated the variables chosen to train the network to predict the launch pad winds. The wind was transformed to u and v components so it could be treated as two scalars. This circumvented the 0-360 discontinuity and provided the neural network with continuous measurements. Subtracting components of the wind at different heights developed measures of vertical wind shear. Likewise, subtracting components of the wind at the same height, but different locations developed horizontal wind shear measurements. Wind component subtractions from different levels of different towers represented a combination of vertical and horizontal shear. The WINDS tower network provided peak wind speeds, and a measure of wind directional deviation that indicates its variability. Sea level pressure observations were also included to describe the wind field.

The remaining variables were largely measures of differential heating. This group contained diverse variables. Temperature, dew point, relative humidity, and cloud ceiling heights were included to aid the neural network in recognizing frontal weather. Vertical and horizontal gradients of temperature and moisture were developed through component subtractions. The solar elevation angles for Kennedy Space center were calculated and included as well.

In all, 153 experimental variables were developed from the three types of observations. Appendix A contains a complete listing.

3.4 *Reducing the Variables*

A salient set of variables was selected from the original 153 through Greene's signal-to-noise data reduction method (Greene, 1999). The method was performed on MATLAB© with an Elman recursive neural network with fifteen hidden nodes (the code is listed in appendix B).

A multi-processor SGI Power ONYX Infinite Reality Systems computer (provided by the Major Shared Resource Center at Wright-Patterson Air Force Base) was used to run the program. The network was trained to forecast the u and v components and peak speed of the wind fifteen minutes into the future. Table 2 lists the fifteen measurements the network trained to forecast. The network was trained on 408 days of variables (96 samples per day) from the winters of 1995-6, 1996-7, and 1997-8. Observations from 1 November through 10 December 1996 validated the network solution. Due to the size of the data set, the least useful variables were initially eliminated ten at a time. As the set became smaller, fewer variables were eliminated each time. Prior values of the predicted values were kept in the training set for programming ease until twenty-five variables remained.

The least predictive variables (at a 15-minute prediction interval) were the upper-air, surface, and buoy variables. The cloud ceiling heights were the least predictive at all. Dew points, upper-air temperatures, and sea level pressures tended to provide little predictive power

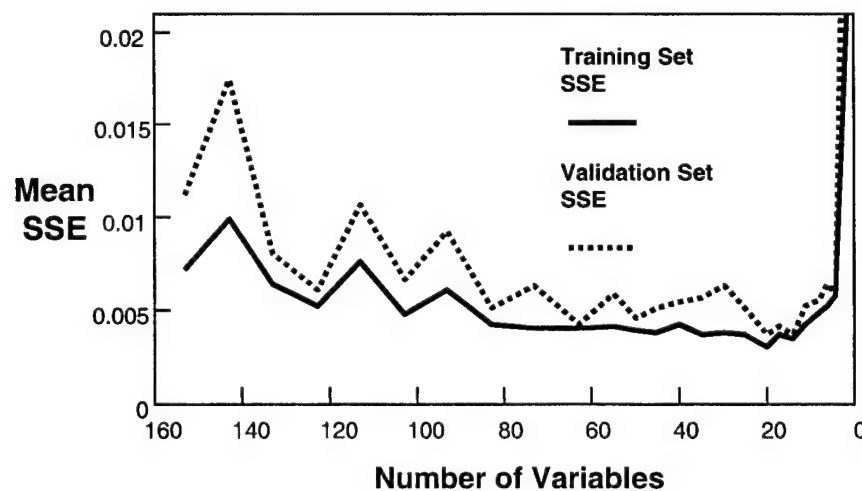


Figure 18 Mean sum of squared error for the network tended to decrease as the least predictive variables were removed. The best performance occurred with 20 variables; removing additional variables resulted in a loss of performance measured by higher mean SSE.

to the neural network. Somewhat surprisingly, previous peak wind speeds also were found not highly predictive of future peak wind speeds.

Figure 18 graphs the values of SSE against the number of variables. The error decreased rather steadily until the minimum occurred at twenty variables. As the set was decreased to one variable, the SSE increased significantly. Of the twenty variables that accomplished the lowest SSE, fifteen were the present values of the predicted variables (Table 2). Of the remaining five, two were a horizontal shear measurements, one a vertical shear measurement, and two were v wind components at 204 feet. Other variables that proved relatively predictive included directional deviation of the wind, the combined horizontal and vertical shear variables, and the u component of the wind at Jacksonville. Table 3 lists these variables. Appendix A includes a more detailed rank order of the variables.

Table 2 The Predicted Variables (Targets)

Tower	Height (ft)	Measurement
0393	60	U wind component
0393	60	V wind component
0393	60	Peak wind speed
0397	60	U wind component
0397	60	V wind component
0397	60	Peak wind speed
1101	54	U wind component
1101	54	V wind component
1101	54	Peak wind speed
1101	162	U wind component
1101	162	V wind component
1101	162	Peak wind speed
0036	90	U wind component
0036	90	V wind component
0036	90	Peak wind speed

Table 3 Ten Additional Predictive Variables

Location	Height (ft)	Variable	Rank
Tower 1101	204	V wind component	7
Towers 1101 & 0061	162	U difference (h. shear)	14
Tower 1101	12 & 162	U difference (h. shear)	15
Tower 3132	204	V wind component	16
Tower 1101	12 & 162	V difference (v. shear)	17
Tower 0393	60	Directional Deviation	21
Towers 1101 & 0714	54	U difference (h. shear)	22
Tower 0061	162	V wind component	23
Tower 3132	394	Directional Deviation	24
Jacksonville	Surface	U wind component	25

3.5 *Training the Network*

Once the most salient set of variables was identified, the random variable was removed, and an Elman Recursive Neural Network was trained to forecast the fifteen target variables fifteen minutes into the future. For experimental purposes, networks were also trained to forecast fewer variables, and predict over longer time intervals, though these models may not have had the most predictive variables for the task. If procedure were followed strictly, one would repeat the signal-to-noise variable screening method to mirror each proposed neural network (it is possible that other variables are more predictive at different time intervals, or more predictive for certain towers). However, given time constraints, the one screening method was used for all forecast times and all fifteen forecast variables.

3.6 *Generating a Forecast Ensemble*

An ensemble of slightly perturbed neural network forecasts was generated through a Fortran program. The program made multiple time-step forecasts by updating input values with time iterated output values. Initial conditions to these forecasts were perturbed using the breeding of growing modes method to develop ensemble forecast members.

3.6.1 *The Basic Model*

The Fortran program used to generate the ensemble of forecasts is listed in appendix B. The program was designed to forecast an array of variables, x_1 through x_n given those variables at a previous time step (Figure 19). One could also include the variables y_1 through y_m to aid prediction (these variables were envisioned as large scale environmental information that does not change significantly over the forecast period). The program passed observed wind variables to the neural network model, and the model forecast the variables one time-step. Then the

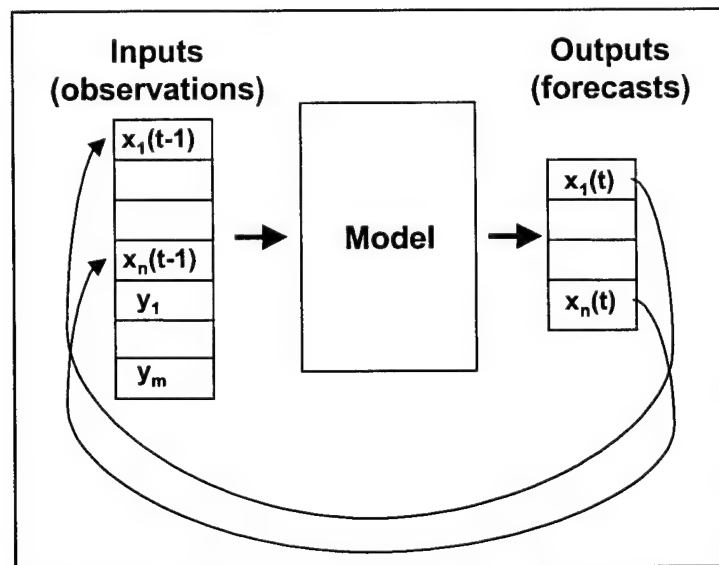


Figure 19 The schematic depicts the basic iterative process of the neural network model. The x values are iterated in time, while the y values remain constant.

program iterated the observations one time-step ahead by replacing the current values with forecast values. The program could run as many time steps as desired. The rationale behind not updating all variables (i.e., the y variables) was so they could be variables that don't change much during the forecast time (such as upper-air or surface observations), and would be difficult to accurately forecast their change on a small scale.

3.6.2 *Perturbing the Model*

To generate an ensemble of slightly different, but equally likely forecasts, the iterative inputs to the model (i.e., the x variables in Figure 19) were perturbed by the breeding of growing modes method. Figure 20 depicts the structure of the growing mode cycle. On the far left of the image, the variables observed at one time step before the present are added to corresponding perturbations. Initially, the perturbations are randomly chosen numbers scaled back to a predefined vector length (this length is chosen to approximate the amount of error in the measurements). The perturbed observations are passed to the model, which produces a forecast for the present time. The forecast for the present time is compared to the present

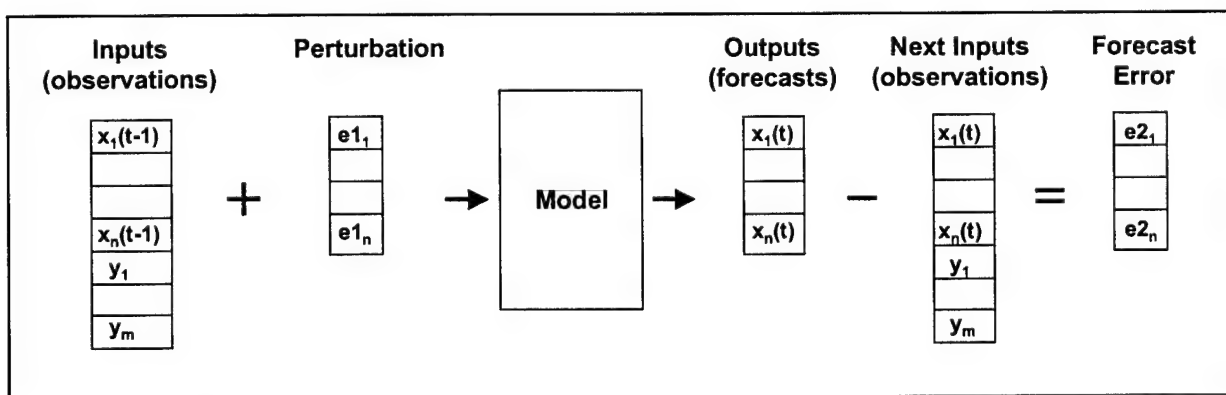


Figure 20 The schematic depicts steps in developing a perturbed forecast representative of a growing mode. The input vector (t-1) is perturbed with an error vector (initially random numbers; subsequently the previous model run's forecast error vector). The perturbed observations (t-1) are passed to the neural network model and produce forecasts. A forecast error vector is found by comparing the forecasts to the current measured values (t). This error vector is used to perturb the current observations, and the forecast runs multiple time-steps.

observations; their difference is the forecast error. This error vector is scaled-back to the predefined error length (as before) and used to perturb the present observations. The program then forecasts multiple time steps based on the present perturbed observations. The cycle repeats, but the scaled back forecast error vector is used to perturb the observations instead of random numbers. By starting with multiple perturbations and modeling each set of perturbed observations separately, an ensemble of forecasts is generated.

4. Results

4.1 Overview

Neural networks were able to achieve skill against persistence. The best results were obtained by limiting the number of tower locations predicted, and increasing the time interval of the prediction.

4.2 Recursive Neural Network Performance

Unfortunately, the MATLAB program did not simulate the trained network as an Elman Neural Network, and simulated it as a feed-forward network instead. This effectively eliminated the memory property that often improves performance of the neural network. Reproducing the network with Fortran code and MathCad© identified this problem. The MATLAB simulation of the network could be only be duplicated by omitting contributions from the recursive nodes (the $y(t-1)$ nodes in Figure 11). Engineers at MATLAB have examined the problem and raised it to the authors of the Elman code. As of yet, the cause of the problem remains unknown. To work around this problem, a feed-forward network was substituted for the Elman network in the Fortran program by removing several lines of instruction.

The network solutions were examined for skill against persistence (i.e., skill against forecasting no change in the wind). To determine skill, the mean absolute error for the model's forecast was compared to that of a persistence forecast. Mean absolute error was calculated for persistence and the forecast as

$$MAE_{pers} = \frac{1}{n} \sum_{k=1}^n |o_{k-1} - o_k| \text{ and}$$

$$MAE_{fcst} = \frac{1}{n} \sum_{k=1}^n |y_k - o_k|,$$

where o is the observed value and y is the forecast value over time interval k .

Networks using the twenty most predictive variables to forecast the wind fifteen minutes ahead at the five locations did not achieve skill against persistence. The mean absolute error for each variable was greater than that of not forecasting at all (Table 4). But encouragingly, the

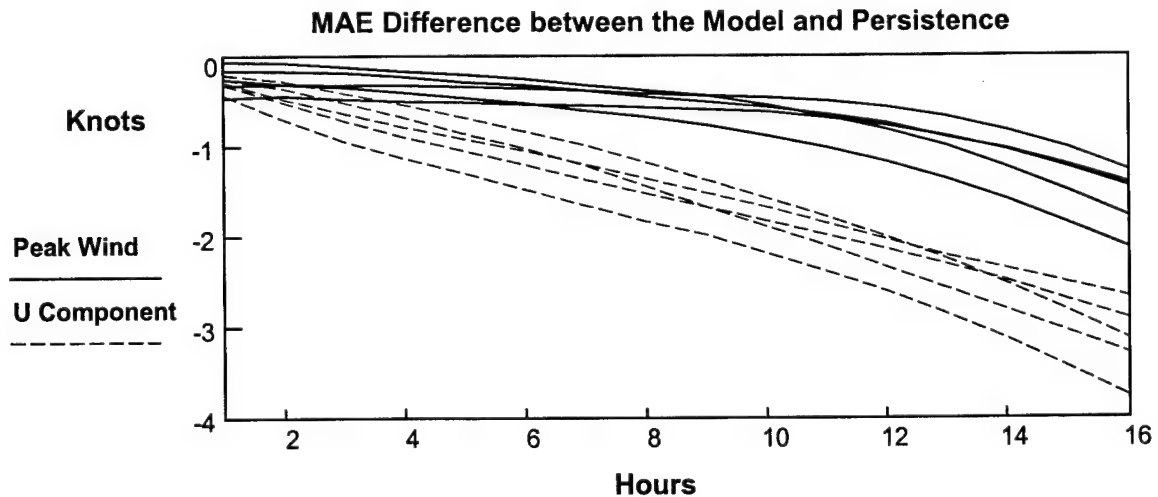


Figure 21 Though none of the variables achieved positive skill against persistence, the peak wind forecasts performed the best.

network tended to produce the least error when predicting peak winds. Figure 21 shows comparisons between the performance of peak wind forecasts and u-wind-component forecasts. Though none achieve a positive difference (representing skill against persistence), the peak wind forecasts performed the best.

Table 4 Mean Absolute Error for a 15-Minute Forecast

Measurement	MAE Forecast	MAE Persistence	Percent error above persistence
0393-60 U	1.19	.88	35%
0393-60 V	1.18	.81	46%
0393-60 PK	1.40	1.17	20%
0397-60 U	1.28	.91	41%
0397-60 V	1.10	.84	31%
0397-60 PK	1.40	1.20	17%
1101-54 U	1.02	.76	34%
1101-54 V	.94	.76	24%
1101-54 PK	1.37	1.10	25%
1101-162 U	1.21	.94	29%
1101-162 V	1.14	.88	30%
1101-162 PK	1.41	1.06	33%
0036-90 U	1.20	.82	46%
0036-90 V	1.48	.82	80%
0036-90 PK	1.53	1.06	44%

Through experimentation, a feed-forward neural network was generated that achieved skill against persistence. The network was designed with twenty-five hidden nodes and used sixty-three input variables. The network was trained to forecast the winds at tower 0393 (a shuttle launch pad) four hours ahead. Thirty-nine days from November and early December 1996 were used to test the network against persistence. Overall, the network outperformed a persistence forecast in absolute error (Table 5). The peak-wind forecast was 11 percent better than persistence on average, and its maximum absolute error was 20 percent less.

Table 5 Absolute Error (knots)

Model Forecast vs. Persistence

	<i>U-Component</i>	<i>V-Component</i>	<i>Peak Wind</i>
Forecast (mean)	3.38	3.04	3.32
Persistence (mean)	3.38	2.95	3.68
Forecast (max)	12.3	18.5	19.7
Persistence (max)	21.7	22.2	24.1

The neural networks were most successful when forecasting the winds for several hours ahead. The networks trained to forecast fifteen minutes ahead never achieved skill against persistence. Generally, forecasts for all time intervals were more similar to persistence than the observed weather. That is, the network tended to forecast the variables to change little. Figure 22 is an example. In the figure, the forecast is more similar to persistence than to the observed weather. But when it varies from persistence, it tends to vary in the direction that gives it predictive power. In the case of the fifteen minute forecasts, the variation from persistence wasn't nearly as predictive. This may be due to measurement resolution. After fifteen minutes,

any one variable changes about one knot on average. The measurement error of the wind sensors is 0.3 m/s, or about 0.6 knots (Computer Sciences Raytheon, 1998). The measurement error, which could account for over half the variation in the wind, may overwhelm the short-term signal.

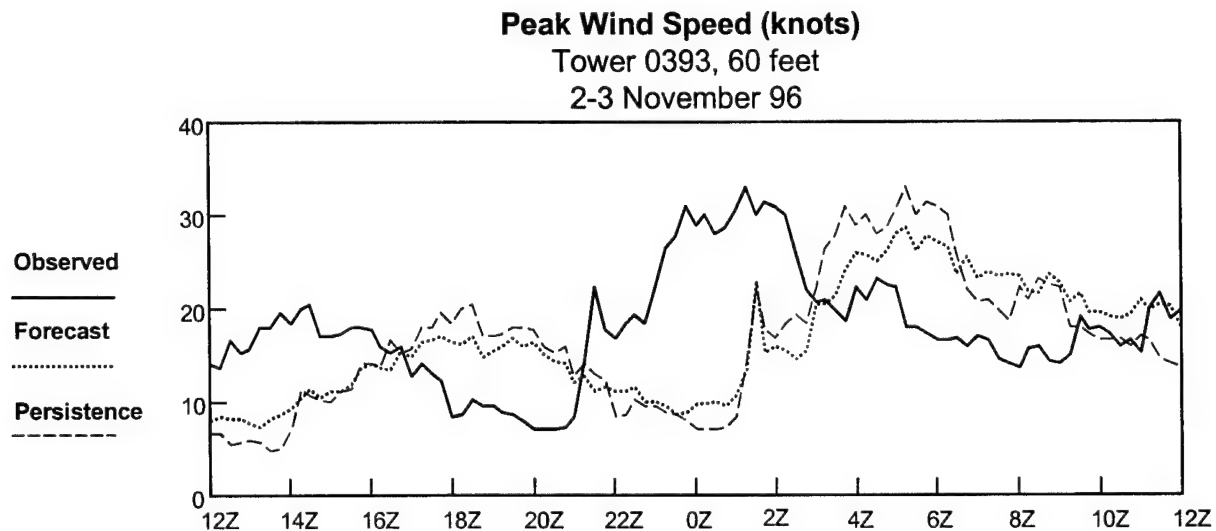


Figure 22 For each 15 minutes, this graph shows the measured peak wind, a four hour forecast, and a four hour persistence forecast (the observed weather lagged four hours). If the forecast were perfect, it would coincide with the solid line. It is much more similar to the persistence forecast, but more often closer to the observed weather. Thus, it achieves skill.

The decision to smooth and down-sample the data may have caused additional error. Smoothing eliminated high frequency signals that may have been predictive, and down-sampling created a small amount of phase error for frequencies less than 15 minutes.

4.3 *Ensemble Model Performance*

The ensemble aspect of this model functioned properly, but did not substantially add to the model's usefulness. The error in the forecasts was overwhelmingly attributable to error in the model, not error in the measurements. Therefore, perturbing the input variables to simulate

variation due to measurement error did not significantly change the forecasts. The end result was the forecasts clustered tightly together and rarely bounded the actual outcome.

The ensemble forecasts diverged after a sufficient number of model iterations. Increasing the magnitude of the perturbation vectors could accelerate divergence. Figure 23 compares perturbed ensemble members to an unperturbed control forecast. The forecast variable and units are unimportant because the model lacked skill, but the paths of the ensemble

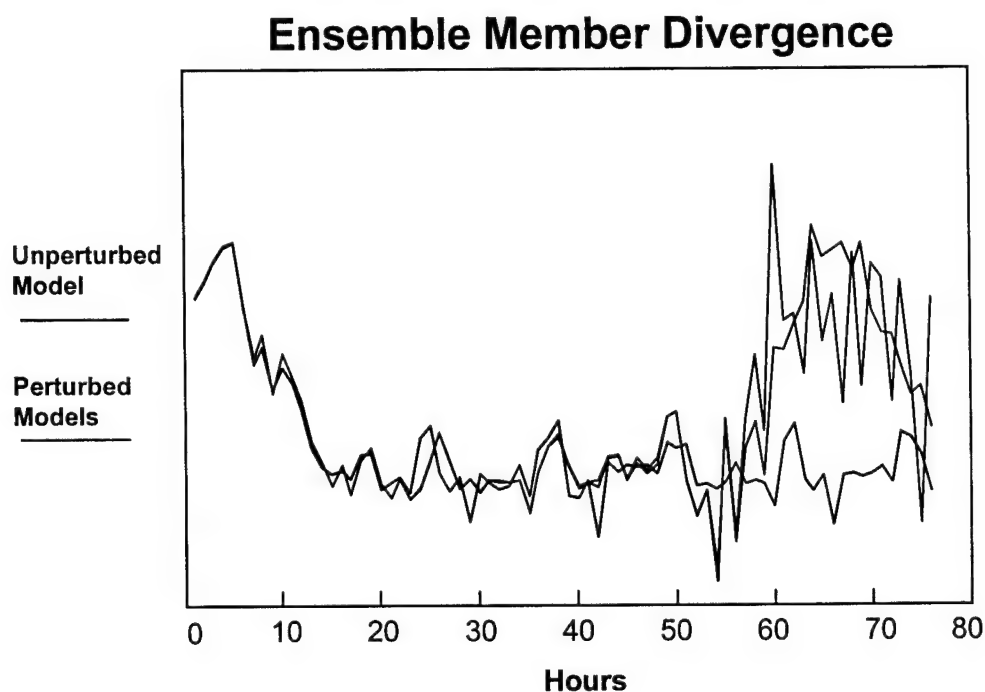


Figure 23 Though the ensemble technique was not as useful as hoped, the technique worked and the perturbed members diverged from the control forecast over time.

members are noteworthy. Initially, the perturbed forecasts are nearly identical to the control forecast. Slowly they begin to diverge, and eventually take different, seemingly non-periodic paths. Appendix C contains the Fortran code for the ensemble model.

5. Conclusions and Recommendations

5.1 *Conclusions*

Neural network models can be trained to skillfully predict wintertime winds at the Kennedy Space Center launch pads. However, the problem is by no means easy, and a training set must be rigorously prepared to achieve an appreciable amount of skill. The amount of skill is also strongly influenced by the neural network architecture, forecast interval, and number of forecast variables. Inevitably, well-trained networks require human-learning as well as machine-learning. A systematic technique that consistently provides the best solution does not exist, so one must learn through trial-and-error. In this thesis, experimentation proved fruitful.

Neural networks performed best on WINDS tower data when trained to forecast periods of several hours. This may be a result of wind sensor measurement error. The sensors' error tolerance is about 0.6 knots, and the mean absolute change in wind at the launch pad towers over fifteen minutes was around 1 knot for the period studied. Over half the variation was attributable to measurement error. Over a period of four hours the mean absolute change in wind rose to 3-4 knots. Only 15-20 percent of the variation could then be attributed to measurement error.

The ensemble aspect of the model functioned properly but did not add value to forecasts. One of the assumptions that must hold true to develop useful ensemble weather forecast is the model must approximate the atmosphere well. Unfortunately, the models did not learn to forecast the weather well enough. The neural network models resembled persistence more than the weather it forecasts. Mean absolute error values confirmed this. An example is the forecast for the peak winds at tower 0393. The mean absolute error between the peak wind forecast and the observed peak wind was 3.32 knots. However, the peak wind forecast was more similar to persistence, and their mean absolute error was only 1.93 knots. The ensemble technique is more

likely to add value if the neural network forecasts are skillful against persistence *and* resemble the observed values more than the persistence values.

5.2 *Recommendations*

The results of this thesis could be improved upon by carefully selecting and screening specialized training data. To aid a neural network in learning wind behavior when the winds are near threshold values, the network should train on those events only. All other weather events may distract the network from learning to predict the near-threshold case. It is important large errors and large changes due to irresolvable processes (i.e., thunderstorms) are screened from the database since they are considered noise to the desired signal. Since the network reduces the sum of *squared* error, large errors seriously distort a neural network's training (Tsoukalas and Uhrig, 1997). High winds due to thunderstorms are particularly distracting to the learning process because their onset is rapid, short-lived, and generally below WINDS network resolution (high winds from a thunderstorm are typically evident at one tower at a time). The network should learn a better solution if it doesn't seek to minimize the large errors due to thunderstorms. Likewise, uninteresting periods of weak winds also may diminish a neural network's ability to predict periods of strong winds and should be eliminated. Since the WINDS tower wind sensors have a fixed error tolerance, ± 0.33 m/s, weak winds may contain a larger percent of error. As the training set becomes more specialized, the network should learn to predict the special situation better.

Another method to improve forecasts is by using a more sophisticated neural network architecture or improved training techniques. The networks in this thesis were effectively trained with feed-forward neural networks. However, recursive neural networks are advantageous because they measure the rate change of input variables over time. Adaptive time delay neural

networks, a specialized form of recursive neural networks, learn the best time intervals to measure the rate change of input variables. These features may reduce a network's sum of squared error performance by as much as an order of magnitude (Gainey, 1993). Many variations in training techniques also exist. All networks developed in this thesis trained with the resilient backpropagation algorithm (a specialized form of the learning algorithm discussed in chapter 3), but many others exist and may train a network more efficiently, if not better.

In conclusion, this thesis demonstrates the feasibility of predicting launch pad winds at KSC/CCAS with a neural network model. Improvements upon these results may be achieved through careful data preparation and successful use of a recursive neural network. A simple feed-forward neural network outperformed a persistence forecast using relatively noisy data. If the data set's noise can be reduced, predictive performance should increase substantially.

APPENDIX A. ORIGINAL MODEL INPUTS

This appendix lists the 153 variables tested for predictive ability towards the winds at towers 0393, 0397, 1101, and 0036. The tower variables are coded as a combination of the tower number and the sensor height in feet. Surface and buoy observations are coded by their name or symbol. Tampa upper-air observations are coded by their pressure level. U = u-wind component, V = v-wind component, PK = peak wind speed, DDEV = directional deviation, T = temperature, Td = dew point temperature, RH = relative humidity, SLP = sea level pressure, CIG = cloud ceiling height, GUST = gust speed, and HT = geopotential height. Rank indicates their relative predictiveness, as determined by the signal-to-noise technique (1 highest, 153 lowest):

Rank	Variable
1	0393-60 U
12	0393-60 V
10	0393-60 PK
8	0397-60 U
13	0397-60 V
3	0397-60 PK
11	1101-54 U
4	1101-54 V
20	1101-54 PK
18	1101-162 U
5	1101-162 V
19	1101-162 PK
6	0036-90 U
2	0036-90 V
9	0036-90 PK
21	0393-60 DDEV
95	0393-60 T
101	0393-60 RH
134	0397-60 DDEV
99	0397-60 T
94	0397-60 RH
	1101-6 T
102	1101-6 RH
	1101-12 U
	1101-12 V
	1101-12 PK
110	1101-12 DDEV
15	1101-12 U - 1101-162 U
17	1101-12 V - 1101-162 V
	1101-54 DDEV
84	1101-54 T
139	1101-54 RH
	1101-204 U
7	1101-204 V
114	1101-204 PK

1101-204 DDEV
 1101-204 U - 1101-54 U
 1101-204 V - 1101-54 V
 1101-162 DDEV
 1101-6 RH - 1101-204 RH
 97 1101-6 T - 1101-204 T
 1101-6 T - 0819-6 T
 92 1101-54 U - 0819-54 U
 85 1101-54 V - 0819-54 V
 119 1101-6 T - 1012-6 T
 1101-54 U - 1012-54 U
 1101-54 V - 1012-54 V
 22 1101-54 U - 0714-54 U
 1101-54 V - 0714-54 V
 1101-54 DDEV - 0714-54 DDEV
 124 0061-6 T
 105 0061-12 U
 0061-12 V
 135 0061-12 PK
 0061-54 U
 0061-54 V
 138 0061-54 PK
 0061-162 U
 23 0061-162 V
 100 0061-162 DDEV
 14 1101-162 U - 0061-162 U
 1101-162 V - 0061-162 V
 1101-162 DDEV - 0061-162 DDEV
 3132-6 T - 0398-6 T
 3132-54 U
 3132-54 V
 115 3132-54 DDEV
 0393-60 U - 3132-54 U
 0393-60 V - 3132-54 V
 0397-60 U - 0415-54 U
 0397-60 V - 0415-54 V
 3132-162 U
 3132-162 V
 125 3132-162 PK
 3132-204 U
 16 3132-204 V
 3132-204 DDEV
 3132-295 U
 3132-295 V
 3132-295 PK
 3132-394 U
 3132-394 V

24	3132-394 DDEV
	3132-492 U
	3132-492 V
	3132-492 PK
	3132-492 U - 3132-54 U
	3132-492 V - 3132-54 V
	3132-492 T - 3132-54 T
	3132-492 U - 0061-54 U
	3132-492 V - 0061-54 V
	3132-54 U - 2016-54 U
123	3132-54 V - 2016-54 V
143	Solar elevation angle
118	Daytona U
86	Daytona V
151	Daytona CIG
	Daytona T
122	Daytona Td
	Daytona SLP
25	Jacksonville U
109	Jacksonville V
153	Jacksonville CIG
96	Jacksonville T
108	Jacksonville Td
	Jacksonville SLP
104	Melborne U
	Melborne V
149	Melborne CIG
106	Melborne T
	Melborne Td
130	Melborne SLP
	Tampa U
137	Tampa V
152	Tampa CIG
121	Tampa T
89	Tampa Td
	Tampa SLP
87	Wpalmbeach U
98	Wpalmbeach V
150	Wpalmbeach CIG
111	Wpalmbeach T
107	Wpalmbeach Td
132	Wpalmbeach SLP
91	41009 U
127	41009 V
116	41009 GUST
113	41009 SLP
126	41009 T

88	SPGF1 U
146	SPGF1 V
133	SPGF1 GUST
117	SPGF1 SLP
	SPGF1 T
147	500mb HT (m)
148	500mb T(C)
	500mb U
136	500mb V
145	700mb T
129	700mb U
128	700mb V
142	850mb T
120	850mb U
112	850mb V
	925mb T
141	925mb Td
103	925mb U
93	925mb V
144	500mb U - 850mb U
131	500mb V - 850mb V
	925mb T - 850mb T
	925mb U - 850mb U
140	925mb V - 850mb V

APPENDIX B. MATLAB CODE

```
% script M-file elmansnr.m
% This script M-file trains an elman neural network on a set of features
% and a deliberately random variable. The first layer weights are
% compared to the weights of the random variable and rank ordered.
% It records the average sum of squared error of the network.
[r1,c1]=size(matrix1);
[r2,c2]=size(matrix2);
r2=r2+1;
matrix3=[matrix1,matrix2];
clear matrix1 matrix2;
[pn,minp,maxp,tn,mint,maxt]=premnmx(matrix3,matrix3(1:15,:));
clear matrix3
noise1=rand(c1-1)';
P=[noise1;pn(:,1:(c1-1))];
clear noise1
T=tn(:,2:(c1));
r1=r1+1;
PR=[];
PRadd=[-1 1];
for num=1:r1
    PR=[PR;PRadd];
end
clear PRadd num
net1=newelm(PR,[15,15],{'tansig','purelin'},'trainrp');
noise2=rand(c2-1)';
v.P=[noise2;pn(:,(c1+1):(c1+c2-1))];
v.T=tn(:,(c1+2):(c1+c2));
clear noise2 pn tn maxp minp maxt mint
net1.trainParam.epochs=200;
net1.trainParam.show=5;
net1=init(net1);
net1.IW{1,1}=net1.IW{1,1}/10;
[net1,tr1]=train(net1,P,T,[],[],v);
clear P T v PR
W1=net1.IW{1,1};
W1sq=W1.*W1;
sumW1sq=sum(W1sq);
for num=1:r1
    sig(num)=10*log10(sumW1sq(num)/sumW1sq(1));
end
[signal order]=sort(sig);
clear sig
signal=[signal;order];
clear order r1 r2 c1 c2 num
```

APPENDIX C. ENSEMBLE MODEL FORTRAN CODE

```

PROGRAM emodel.f
*****
* This program uses Elman RNN weight and bias matrices to produce an
* ensemble of wind forecasts for the Kennedy Space Center.
* This model forecasts at one hour intervals.
* MATLAB seemed to train and simulate the network as a feed-forward
* network. Therefore I've commented out the recursive matrix.
*
* Variables
* meas is a source matrix of measurements for this program.
* le is the number of time steps read into the source matrix.
* inp is the number of model inputs (and variables in meas).
* outp is the number of outputs in the forecast.
* subm is the subset of meas the ensemble model uses.
* oldm is the set of measurements from the past time step.
* newm is the set of measurements from the present time step.
* mod is the number of models, the first model is unperturbed.
* nodes is the number of hidden nodes in the Elman RNN.
* step is the number of time steps the model runs.
* run is the number of consecutive periods the model runs.
* pert is the matrix of perturbations.
* randv is a matrix of random variables to initialize the perturbations.
* minm and maxm are vectors of minimum and maximum measurements for
* each variable. They are taken from the training set of data.
* oldh is a matrix of hidden nodes from the previous time step.
* warm is the number of time periods the model warms up its growing
* modes and hidden nodes before making a forecast.
* w1 is the matrix of weights between the inputs and hidden nodes.
* w1r is the matrix of weights between the past hidden nodes and
* the present hidden nodes.
* b1 is the vector of biases to the hidden nodes.
* w2 is the matrix of weights between the hidden nodes and the outputs.
* b2 is the vector of biases to the outputs.
* maxerror is the maximum allowed magnitude of the growing modes.
* cump (cumulative perturbation) is used to find the perturbations'
* magnitudes.
* start chooses the starting location of subm within meas.
* i,l,m,r and t are counting variables for loops.
* a and b are used loosely as counters for both outp and nodes.

      parameter (mod=3,inp=15,step=8,le=500,run=3)
      parameter (warm=10,outp=15,nodes=15)
      parameter (size=le+run+warm+4)
      real meas(le,inp),subm(size,inp),oldm(inp),newm(inp)
      real pert(mod,inp),fcst2(mod,outp),fcst3(mod,outp,step,run)
      real randv(mod,inp),minm(inp),maxm(inp),oldh(mod,nodes)
      real w1(inp,nodes),w1r(nodes,nodes),b1(nodes)
      real w2(nodes,outp),b2(outp),maxerror,cump
      integer m,i,t,l,r,start,a,b

* Set constants
      start=1
      maxerror=.15

```



```

* Output files
  open(unit=11,file='mod1.dat')
  open(unit=12,file='mod2.dat')

* Read data
  open (20, file='./vars15.txt')
  read(20,*) ((meas(l,i),i=1,inp),l=1,le)
  open (25, file='./w12.txt')
  read(25,*) ((w1(i,a),i=1,inp),a=1,nodes)
  open (30, file='./w1r2.txt')
  read(30,*) ((w1r(a,b),a=1,nodes),b=1,nodes)
  open (35, file='./w22.txt')
  read(35,*) ((w2(a,b),a=1,nodes),b=1,outp)
  open (40, file='./b12.txt')
  read(40,*) (b1(a),a=1,nodes)
  open (45, file='./b22.txt')
  read(45,*) (b2(a),a=1,outp)
  open (50, file='./randv.txt')
  read(50,*) ((randv(m,i),i=1,inp),m=1,mod)
  open (55, file='./minm2.txt')
  read(55,*) (minm(i),i=1,inp)
  open (60, file='./maxm2.txt')
  read(60,*) (maxm(i),i=1,inp)

* Initialize perturbations.
* Fill the perturbation matrix with zeroes. Leave zeroes in the
* first row so the first model is unperturbed.
* Scale back random vectors magnitude "maxerror"
  do m=1,mod
    do i=1,inp
      pert(m,i)=0
    enddo
  enddo
  do m=2,mod
    do a=1,outp
      pert(m,a)=randv(m,a)*10
    enddo
  enddo
  do m=2,mod
    cump=0
    do a=1,outp
      cump=cump+pert(m,a)**2
    enddo
    if(cump.gt.maxerror)then
      do a=1,outp
        pert(m,a)=pert(m,a)*maxerror/cump
      enddo
    end if
  enddo

* Initialize oldh to zeroes.
  do a=1,nodes
    do m=1,mod
      oldh(m,a)=0
    enddo
  enddo

```

```

* Choose the data and normalize between 1 and -1.
  do a=1,step+run+warm
    do i=1,inp
      subm(a,i)=2*(meas(a-1+start,i)-minm(i))/(maxm(i)-minm(i))-1
    enddo
  enddo

* Assign values to oldm and newm.
  do i=1,inp
    oldm(i)=subm(1,i)
    newm(i)=subm(2,i)
  enddo

* Perturb measurements.
  call modpert(oldm,pert)

* Run model
  call model(pert,oldh,fcst2,w1,w1r,w2,b1,b2)
*   print*, 'fcsts ',fcst2(1,1),fcst2(1,2),fcst2(1,3)

* Find new perturbations.
  call newpert(newm,fcst2,pert,maxerror)

* Warm-up recursive hidden-nodes and growing modes (perturbations).
  do a=2,warm
    do i=1,inp
      oldm(i)=subm(a,i)
      newm(i)=subm(a+1,i)
    enddo
    call modpert(oldm,pert)
    call model(pert,oldh,fcst2,w1,w1r,w2,b1,b2)
  *   print*, 'fcsts ',fcst2(1,1),fcst2(1,2),fcst2(1,3)
  call newpert(newm,fcst2,pert,maxerror)
  enddo

* Generate "run" forecasts of "step" length.
  b=warm-1
  do r=1,run
    b=b+1
    do i=1,inp
      oldm(i)=subm(b,i)
      newm(i)=subm(b+1,i)
    enddo
    call modpert(oldm,pert)
    call model(pert,oldh,fcst2,w1,w1r,w2,b1,b2)
    call newpert(newm,fcst2,pert,maxerror)
    call modpert(newm,pert)
    do t=1,step
      call model(pert,oldh,fcst2,w1,w1r,w2,b1,b2)
      do m=1,mod
        do a=1,outp
          pert(m,a)=fcst2(m,a)
          fcst3(m,a,t,r)=0.5*(fcst2(m,a)+1)*
$                                (maxm(a)-minm(a))+minm(a)
          enddo
        enddo
      enddo
    enddo
  enddo

```

```

        do m=1,mod
            do t=1,step
                print*,fcst3(m,1,t,r),fcst3(m,2,t,r),fcst3(m,3,t,r),
$                m,'m',t,'t',r,'r'
            enddo
        enddo
    enddo
enddo

* Write to output file

        write(11,21)((fcst3(1,a,t,1),a=1,oupt),t=1,step)
        write(12,21)((fcst3(2,a,t,1),a=1,oupt),t=1,step)
21    format(15f9.3)
        end

*****
* model subroutine
*****
        subroutine model(pert,oldh,fcst,w1,w1r,w2,b1,b2)

* Declare arrays and variables
        parameter(mod=3,inp=15,oupt=15,nodes=15)
        integer a,b,m,i
        real w1(inp,nodes),b1(nodes),w1r(nodes,nodes)
        real w2(nodes,oupt),b2(oupt),pert(mod,inp),fcst(mod,oupt)
        real hidn(nodes),nin(nodes),nout(oupt),oldh(mod,nodes)
* The first subscript in w1r is for the old hidden nodes. The
* second subscript indicates the current hidden node it connects to.

* Simulate Elman RNN
        do m=1,mod
            do a=1,nodes
                nin(a)=0
                do i=1,inp
                    nin(a)=nin(a)+w1(i,a)*pert(m,i)
                enddo
*                do b=1,nodes
*                    nin(a)=nin(a)+w1r(b,a)*oldh(m,a)
*                enddo
                hidn(a)=(exp(nin(a)+b1(a))-exp(-nin(a)-b1(a)))/
$                (exp(nin(a)+b1(a))+exp(-nin(a)-b1(a)))
*                oldh(m,a)=hidn(a)
            enddo
            do a=1,oupt
                nout(a)=0
                do b=1,nodes
                    nout(a)=nout(a)+w2(b,a)*hidn(b)
                enddo
                fcst(m,a)=nout(a)+b2(a)
            enddo
        enddo
    enddo
end

```

```

*****
*  modpert subroutine                                     *
*****

      subroutine modpert(data,pert)

*  Declare arrays and variables
      parameter (mod=3,inp=15,outp=15)
      integer m,i
      real data(inp),pert(mod,inp)

*  Perturb measurements
      do m=1,mod
        do i=1,outp
          pert(m,i)=pert(m,i)+data(i)
        enddo
        do i=outp+1,inp
          pert(m,i)=data(i)
        enddo
      enddo
      end

*****
*  newpert subroutine                                     *
*****

      subroutine newpert(newm,fcst,pert,maxerror)

*  Declare arrays and variables
      parameter (mod=3,inp=15,outp=15)
      integer a,m
      real newm(inp),fcst(mod,outp),pert(mod,inp),maxerror,cump

*  Calculate new perturbations
      do a=1,outp
        pert(1,a)=0
      enddo
      do m=2,mod
        do a=1,outp
          pert(m,a)=fcst(m,a)-newm(a)
        enddo
      enddo

*  Scale new perturbations
      do m=2,mod
        cump=0
        do a=1,outp
          cump=cump+pert(m,a)**2
        enddo
        if(cump.gt.maxerror) then
          do a=1,outp
            pert(m,a)=pert(m,a)*maxerror/cump
          enddo
        end if
      enddo
      end

```

BIBLIOGRAPHY

- Bishop, Christopher M., 1995. *Neural Networks for Pattern Recognition*. Oxford: Clarendon Press, 482 pp.
- Computer Sciences Raytheon, Range Technical Services, 1998. *Eastern Range Instrument Handbook*, Revision 8, 10.5-1-10.5-20 pp.
- Demuth, Howard and Mark Beale, 1998. *Neural Network Toolbox User's Guide*. Boston: The Mathworks, Inc., 695 pp.
- Fugita, T. T., 1986. "Mesoscale Classifications: Their History and Their Application to Forecasting" in *Mesoscale Meteorology and Forecasting*. Ed. Peter S. Ray. Boston: American Meteorological Society, 18-35 pp.
- Gainey, James C., 1993. *Predicting Nonlinear Time Series*. MS Thesis, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH. 123 pp.
- Gershenfeld, Neil A., 1999. *The Nature of Mathematical Modeling*. Cambridge: Cambridge University Press, 344 pp.
- Gershenfeld, Neil A. and Andreas S. Weigend, 1994. "The Future of Time Series: Learning and Understanding" in *Time Series Prediction: Forecasting the Future and Understanding the Past*. Ed. Andreas S. Weigend and Neil A. Gershenfeld. Reading MA: Addison-Wesley Publishing Company, 1-70 pp.
- Greene, Kelly A., 1998. *Feature Saliency in Artificial Neural Networks with Application to Modeling Workload*. Phd dissertation, AFIT/DS/ENS/98-02. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH. 296 pp.
- Haltiner, George J., and Roger Terry Williams, 1980. *Numerical Prediction and Dynamic Meteorology, Second Edition*. New York: John Wiley & Sons, Inc., 477 pp.
- Haykin, Simon, 1994. *Neural Networks, A Comprehensive Guide*. New York: Macmillan College Publishing Company, 696 pp.
- Hess, Seymour L., 1959. *Introduction to Theoretical Meteorology*. Malabar, FL: Robert E. Krieger Publishing Company, Inc., 362 pp.
- Holton, James R., 1992. *An Introduction to Dynamic Meteorology*. San Diego: Academic Press, Inc., 511 pp.

- Hsieh, William W. and Benyang Tang, 1998. Applying neural network models to prediction and data analysis in meteorology and oceanography. *Bulletin of the American Meteorological Society*, **79**, 1855-1870 pp.
- Lorenz, Edward N., 1963. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, **20**, 130-141 pp.
- Lorenz, Edward N., 1993. *The Essence of Chaos*. Seattle: University of Washington Press, 227 pp.
- Presidential Commission on the Space Shuttle Accident, 1986. *Report of the Presidential Commission on the Space Shuttle Challenger Accident*. Washington: Government Printing Office, 255 pp.
- Roeder, William, 25 August 1998. Chief of Operations Support Flight and Science and Technical Training Officer for the 45th Weather Squadron, Patrick Air Force Base, FL. Personal interview.
- Toth, Zoltan and Kalnay, Eugenia, 1993. Ensemble forecasting at NMC: the generation of perturbations. *Bulletin of the American Meteorological Society*, **74**, 2317-2330 pp.
- Tsoukalas, Lefteri H. and Uhrig, Robert E., 1997. *Fuzzy and Neural Approaches in Engineering*. New York: John Wiley & Sons, Inc., 587 pp.
- Weiss, Sholom M. and Kulikowski, Casimir A. 1991. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. San Mateo, CA: Morgan Kaufman Publishers, Inc., 223 pp.
- Wilks, D. S., 1995. *Statistical Methods in the Atmospheric Sciences*. San Diego: Academic Press, Inc., 467 pp.

VITA

Captain Steven J. Storch grew up in Madison, Wisconsin, and graduated from LaFollette High School. He attended the University of Wisconsin - Madison, where he graduated with a degree in meteorology. His first assignment was to Howard AFB, Panama, in February 1993. At Howard he forecasted cloud-cover amounts for reconnaissance flights involved in the drug war. He also worked as a staff officer for USSOUTHCOM and provided briefing support to the CINCSOUTH. His second assignment was to the 52nd Fighter Wing at Spangdahlem AB, Germany, in March 1995. He was the weather flight's instructor meteorologist at Spangdahlem. He entered the Graduate School of Engineering, Air Force Institute of Technology, in August 1997. Captain Storch enjoys his friendships, cycling, and public radio.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE Mar 1999		3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE Predicting Launch Pad Winds at the Kennedy Space Center With a Neural Network Model			5. FUNDING NUMBERS	
6. AUTHOR(S) Steven J. Storch, Capt, USAF				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AFIT/ENP 2950 P. Street Wright-Patterson AFB, OH 45433 Attn: LtCol Cecilia Miner COMM: (937) 255-3636 DSN: 785-3636 cminer@afit.af.mil			8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/ENP/99M-11	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) 45 WS/SYR 1201 Minuteman Street Patrick AFB, FL 32925-3238 Attn: Mr. William Roeder COMM: (407) 853-8410 DSN: 467-8410 william-roeder@pafb.af.mil			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for public release; distribution unlimited			12b. DISTRIBUTION CODE A	
13. ABSTRACT (Maximum 200 words) <p>This thesis uses neural networks to forecast winds at the Kennedy Space Center and the Cape Canaveral Air Station launch pads. Variables are developed from WINDS tower observations, surface and buoy observations, and an upper-air sounding. From these variables, a smaller set of predictive inputs is chosen using a signal-to-noise variable screening method. A neural network is then trained to forecast launch pad winds from the inputs. The network forecasts are compared to persistence, and peak wind predictions are found skillful compared to persistence.</p> <p>An ensemble modeling technique using Toth's and Kalnay's breeding of growing modes method is explored with neural networks. The inputs are perturbed an amount representative of measurement error. Ensemble member forecasts are found to diverge, but the ensemble spread does not often encompass the resulting weather. This is due to a disproportionate amount of error originating from the model compared to error originating from measurements.</p>				
14. SUBJECT TERMS neural network, model, wind, Florida, ensemble			15. NUMBER OF PAGES 71	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	